In [98]:
```python
# Project Name: Loan Prediction System

# Project Demonstration:
# Loading & Exploring the Data
# Working With Missing Values
# Dropping Unecessary Columns
# Visualization For Data Presentation
# Encoding the Categorical Data
# Model Deployment-:
#     Data Segregation
#     Using GaussianNB
#     Loss Function
#     Using SVC with grid search CV
#     XGBoost Classifier
#     Decision tree using randomized search
#     Random forest using randomized search
#     saving the model
#     coding a web app that will take user input and predict chances of user getting loan


# Hello Folks, My name is Nirbhay Tiwari I am Data Scientist at NimbusPost, In this Project We will see How we Deal with Classification Problem using Supervised Algorithm
```

In [99]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [100…
```python
train_data = pd.read_csv("loan_data_set.csv")
```

In [101…
```python
train_data.head()
```

Out[101]:

	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	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

In [102…
```python
# lets check the shape of the data

train_data.shape
```

Out[102]:
```
(614, 13)
```

In [103…
```python
train_data.describe(include='all')
```

Out[103]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
count	614	601	611	599	614	582	614.000000	614.000000	592.000000	600.00000	564.000000	614	614
unique	614	2	2	4	2	2	NaN	NaN	NaN	NaN	NaN	3	2
top	LP001002	Male	Yes	0	Graduate	No	NaN	NaN	NaN	NaN	NaN	Semiurban	Y
freq	1	489	398	345	480	500	NaN	NaN	NaN	NaN	NaN	233	422
mean	NaN	NaN	NaN	NaN	NaN	NaN	5403.459283	1621.245798	146.412162	342.00000	0.842199	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	6109.041673	2926.248369	85.587325	65.12041	0.364878	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	150.000000	0.000000	9.000000	12.00000	0.000000	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	2877.500000	0.000000	100.000000	360.00000	1.000000	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	3812.500000	1188.500000	128.000000	360.00000	1.000000	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	5795.000000	2297.250000	168.000000	360.00000	1.000000	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	81000.000000	41667.000000	700.000000	480.00000	1.000000	NaN	NaN

In [104…
```python
train_data.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         592 non-null    float64
 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    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [105…
```python
def missing_values(df):
    a = num_null_values = df.isnull().sum()
    return a
```

In [106…
```python
missing_values(train_data)
```

Out[106]:
```
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 [107…
```python
train_data.drop(["Loan_ID", "Dependents"], axis=1, inplace =True)
```

In [108…
```python
train_data
```

Out[108]:

	Gender	Married	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Male	No	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	Male	Yes	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Male	Yes	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Male	Yes	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	Male	No	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...
609	Female	No	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	Male	Yes	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	Male	Yes	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	Male	Yes	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	Female	No	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

614 rows × 11 columns

In [109…
```python
# Dealing with Categorical Missing Values Replacing All the missing category with most frequent value using mode function

cols = train_data[["Gender","Married","Self_Employed"]]

for i in cols:
      train_data[i].fillna(train_data[i].mode().iloc[0], inplace = True)
```

In [110…
```python
train_data.isnull().sum()
```

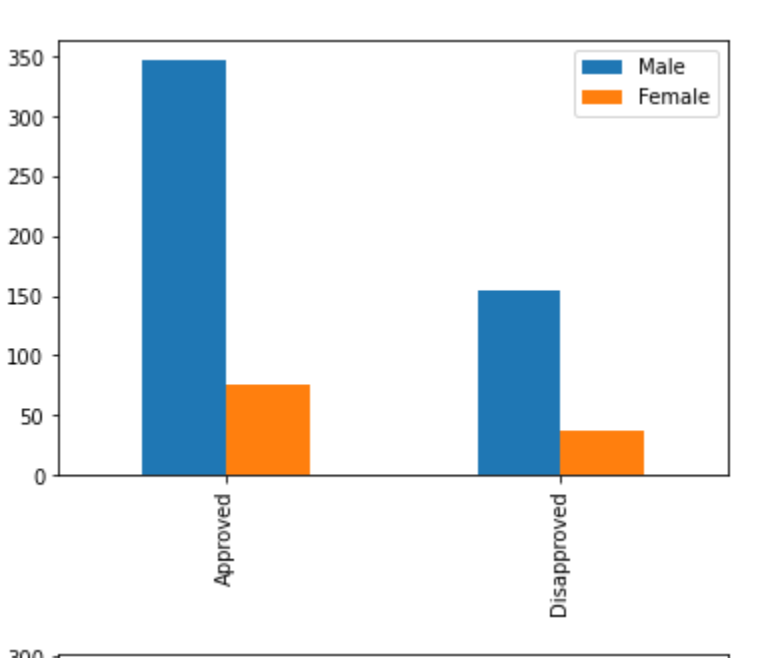
Out[110]:
```
Gender                0
Married               0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

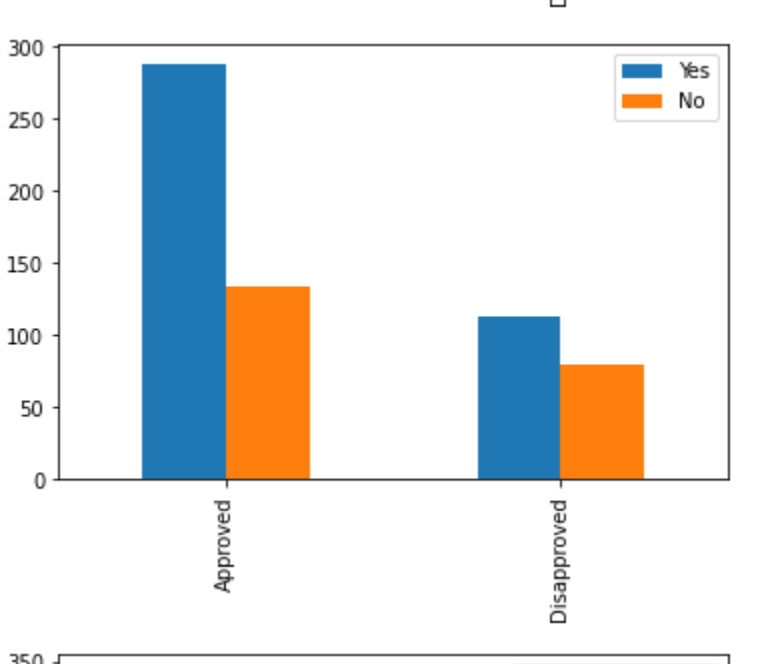
In [111…
```python
# Dealing with Numerical Missing Values Replacing all missing values with mean value


n_cols = train_data[["LoanAmount","Loan_Amount_Term","Credit_History"]]

for i in n_cols:
    train_data[i].fillna(train_data[i].mean(axis=0),inplace=True)
```

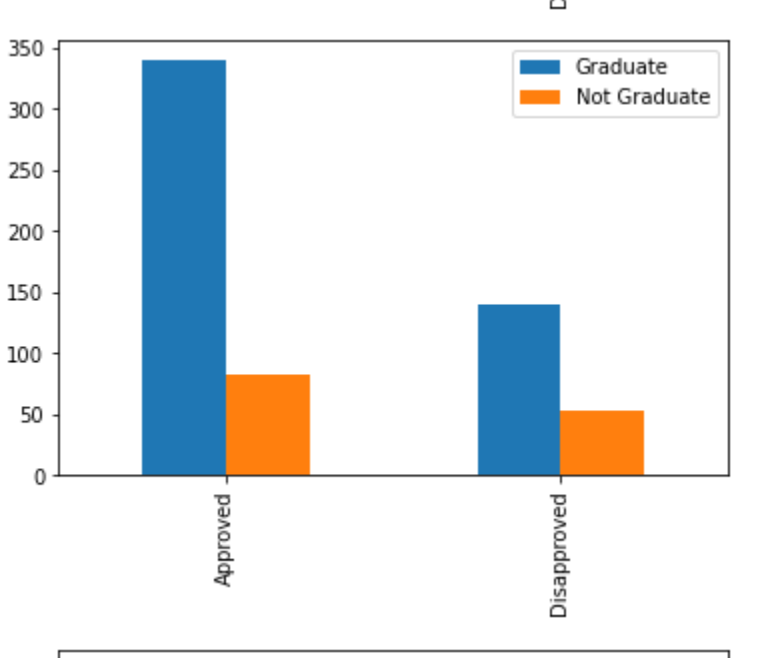
In [112…
```python
# Now we will create a function that will take paramter and show barchart

def bar_chart(col):
    Approved = train_data[train_data["Loan_Status"]=="Y"][col].value_counts()
    Disapproved = train_data[train_data["Loan_Status"]=="N"][col].value_counts()
    df1 = pd.DataFrame([Approved, Disapproved])
    df1.index = ["Approved", "Disapproved"]
    df1.plot(kind="bar")
```

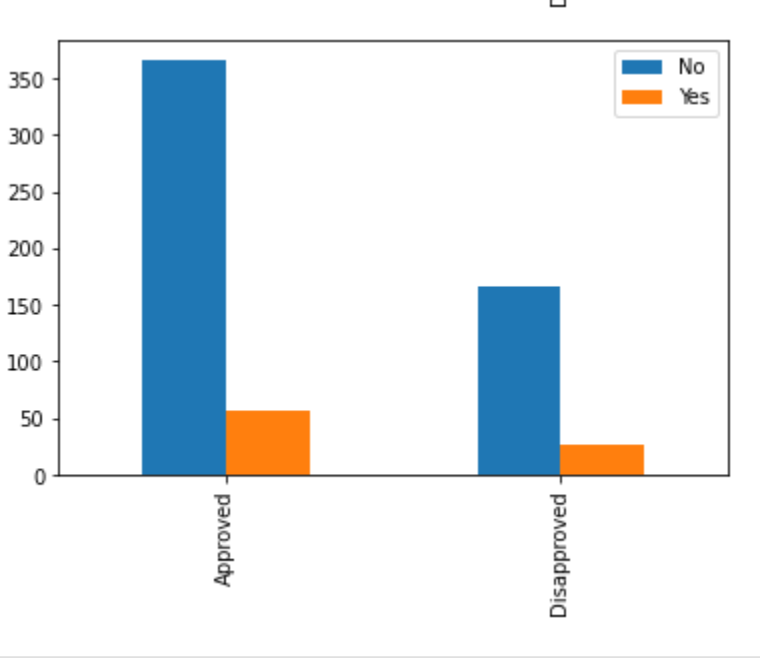
In [113…
```python
bar_chart("Gender"), bar_chart("Married"), bar_chart("Education"), bar_chart("Self_Employed")
```

Out[113]:
```
(None, None, None, None)
```









In [114…
```python
# Lets Encode the gategorical columns value by assigning a numeric identifier to each of them

from sklearn.preprocessing import OrdinalEncoder

ord_enc = OrdinalEncoder()
train_data[['Gender','Married','Education','Self_Employed','Property_Area','Loan_Status']] = ord_enc.fit_transform(train_data[['Gender','Married','Education','Self_Employed','Property_Area','Loan_Status']])
```

In [115…
```python
train_data.head()
```

Out[115]:

	Gender	Married	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	1.0	0.0	0.0	0.0	5849	0.0	146.412162	360.0	1.0	2.0	1.0
1	1.0	1.0	0.0	0.0	4583	1508.0	128.000000	360.0	1.0	0.0	0.0
2	1.0	1.0	0.0	1.0	3000	0.0	66.000000	360.0	1.0	2.0	1.0
3	1.0	1.0	1.0	0.0	2583	2358.0	120.000000	360.0	1.0	2.0	1.0
4	1.0	0.0	0.0	0.0	6000	0.0	141.000000	360.0	1.0	2.0	1.0

In [116…
```python
# Now lets change the decimal value with whole number by converting the datatype as integer

train_data[['Gender','Married','Education','Self_Employed','Property_Area','Loan_Status']] = train_data[['Gender','Married','Education','Self_Employed','Property_Area','Loan_Status']].astype('int64')
```

In [117…
```python
train_data
```

Out[117]:

	Gender	Married	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	1	0	0	0	5849	0.0	146.412162	360.0	1.0	2	1
1	1	1	0	0	4583	1508.0	128.000000	360.0	1.0	0	0
2	1	1	0	1	3000	0.0	66.000000	360.0	1.0	2	1
3	1	1	1	0	2583	2358.0	120.000000	360.0	1.0	2	1
4	1	0	0	0	6000	0.0	141.000000	360.0	1.0	2	1
...	...	...	...	...	...	...	...	...	...	...	...
609	0	0	0	0	2900	0.0	71.000000	360.0	1.0	0	1
610	1	1	0	0	4106	0.0	40.000000	180.0	1.0	0	1
611	1	1	0	0	8072	240.0	253.000000	360.0	1.0	2	1
612	1	1	0	0	7583	0.0	187.000000	360.0	1.0	2	1
613	0	0	0	1	4583	0.0	133.000000	360.0	0.0	1	0

614 rows × 11 columns

In [118…
```python
from sklearn.model_selection import train_test_split

X = train_data.drop("Loan_Status", axis=1)
y = train_data["Loan_Status"]

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=2)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(429, 10)
(429,)
(185, 10)
(185,)
```

In [119…
```python
from sklearn.naive_bayes import GaussianNB

gfc = GaussianNB()
gfc.fit(X_train, y_train)
pred1 = gfc.predict(X_test)
```

In [120…
```python
from sklearn.metrics import precision_score, recall_score, accuracy_score

def loss(y_true, y_pred):
    pre = precision_score(y_true, y_pred)
    rec = recall_score(y_true, y_pred)
    acc = accuracy_score(y_true, y_pred)

    print(pre)
    print(rec)
    print(acc)
```

In [121…
```python
loss(y_test, pred1)
```

```
0.7834394904458599
0.968503937007874
0.7945945945945946
```

In [122…
```python
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

param_grid = {'C' : [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}
grid = GridSearchCV(SVC(), param_grid, refit=True, verbose = 3)
grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.674 total time=   0.0s
[CV 3/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.674 total time=   0.0s
[CV 2/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.674 total time=   0.0s
[CV 3/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.674 total time=   0.0s
[CV 4/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.698 total time=   0.0s
[CV 5/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.671 total time=   0.0s
[CV 1/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.663 total time=   0.0s
[CV 2/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.628 total time=   0.0s
[CV 3/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.605 total time=   0.0s
[CV 4/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.605 total time=   0.0s
[CV 5/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.671 total time=   0.0s
[CV 1/5] END .........C=10, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END .........C=10, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END .........C=10, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END .........C=10, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END .........C=10, gamma=1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.674 total time=   0.0s
[CV 3/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.674 total time=   0.0s
[CV 3/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.671 total time=   0.0s
[CV 1/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.674 total time=   0.0s
[CV 2/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.640 total time=   0.0s
[CV 3/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.640 total time=   0.0s
[CV 4/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.570 total time=   0.0s
[CV 5/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.612 total time=   0.0s
[CV 1/5] END ........C=100, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ........C=100, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END ........C=100, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ........C=100, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END ........C=100, gamma=1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.674 total time=   0.0s
[CV 3/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.674 total time=   0.0s
[CV 3/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.698 total time=   0.0s
[CV 5/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.671 total time=   0.0s
[CV 1/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.674 total time=   0.0s
[CV 2/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.640 total time=   0.0s
[CV 3/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.628 total time=   0.0s
[CV 4/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.570 total time=   0.0s
[CV 5/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.612 total time=   0.0s
[CV 1/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 3/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.674 total time=   0.0s
[CV 3/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.686 total time=   0.0s
[CV 5/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=0.694 total time=   0.0s
[CV 1/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 2/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.674 total time=   0.0s
[CV 3/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.686 total time=   0.0s
[CV 4/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.698 total time=   0.0s
[CV 5/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=0.671 total time=   0.0s
[CV 1/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.674 total time=   0.0s
[CV 2/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.640 total time=   0.0s
[CV 3/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.616 total time=   0.0s
[CV 4/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.581 total time=   0.0s
[CV 5/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.635 total time=   0.0s
```

Out[122]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100, 1000],
                         'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
                         'kernel': ['rbf']},
             verbose=3)
```

In [123…
```python
grid.best_params_
```

Out[123]:
```
{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}
```

In [124…
```python
svc = SVC(C= 0.1, gamma= 1, kernel= 'rbf')
svc.fit(X_train, y_train)
pred2 = svc.predict(X_test)
loss(y_test, pred2)
```

```
0.6864864864864865
1.0
0.6864864864864865
```

In [125…
```python
# !pip install xgboost
```

In [126…
```python
from xgboost import XGBClassifier

xgb = XGBClassifier(learning_rate = 0.1,
                    n_estimators = 1000,
                    max_depth = 3,
                    min_child_weight = 1,
                    gamma=0,
                    subsample = 0.8,
                    colsample_bytree=0.8,
                    objective='binary:logistic',
                    nthread=4,
                    scale_pos_weight=1,
                    seed=27)

xgb.fit(X_train, y_train)
pred3 = xgb.predict(X_test)
loss(y_test, pred3)
```

```
0.8161764705882353
0.8740157480314961
0.7783783783783784
```

In [127…
```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV

def randomized_search(params, runs=20, clf=DecisionTreeClassifier(random_state=2)):
    rnd_clf = RandomizedSearchCV(clf, params, n_iter=runs, cv=5, n_jobs=-1, random_state=2)
    rnd_clf.fit(X_train, y_train)
    best_model = rnd_clf.best_estimator_

    #Extract Best Score
    best_score = rnd_clf.best_score_

    # Print Best Score
    print("Training Score : {:.3f}".format(best_score))

    # Predict test Set Labels
    y_pred = best_model.predict(X_test)

    # Compute Accuracy
    accuracy = accuracy_score(y_test, y_pred)

    # Print Accuracy
    print('Test Score: {:.3f}'.format(accuracy))

    return best_model
```

In [128…
```python
randomized_search(params={'criterion': ['entropy', 'gini'],
                          'splitter': ['random', 'best'],
                          'min_weight_fraction_leaf':[0.0, 0.0025, 0.005, 0.0075, 0.01],
                          'min_samples_split':[2, 3, 4, 5, 6, 8, 10],
                          'min_samples_leaf':[1, 0.01, 0.02, 0.03, 0.04],
                          'min_impurity_decrease':[0.0, 0.0005, 0.005, 0.05, 0.10, 0.15, 0.2],
                          'max_leaf_nodes':[10, 15, 20, 25, 30, 35, 40, 45, 50, None],
                          'max_features':['auto',0.95, 0.90, 0.85, 0.80, 0.75, 0.70],
                          'max_depth':[None, 2, 4, 6, 8],
                          'min_weight_fraction_leaf':[0.0, 0.0025, 0.005, 0.0075, 0.01, 0.05]}),
```

```
Training Score : 0.813
Test Score: 0.800
```

Out[128]:
```
(DecisionTreeClassifier(max_depth=8, max_features=0.9, max_leaf_nodes=30,
                        min_impurity_decrease=0.05, min_samples_leaf=0.02,
                        min_samples_split=10, min_weight_fraction_leaf=0.005,
                        random_state=2, splitter='random'),)
```

In [129…
```python
ds = DecisionTreeClassifier(max_depth=8, max_features=0.9, max_leaf_nodes=30,
                        min_impurity_decrease=0.05, min_samples_leaf=0.02,
                        min_samples_split=10, min_weight_fraction_leaf=0.005,
                        random_state=2, splitter='random')

ds.fit(X_train, y_train)
pred4 = ds.predict(X_test)
loss(y_test, pred4)
```

```
0.78125
0.984251968503937
0.8
```

In [130…
```python
# Model Deployment

import joblib
joblib.dump(ds, "model.pkl")
model = joblib.load('model.pkl')
model.predict(X_test)
```

Out[130]:
```
array([0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 1], dtype=int64)
```

In [ ]: