loan_prediction_system

December 17, 2022

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

[ ]:
```
# LOADING AND EXPLORING DATA
```

[ ]:
```
train_data = pd.read_csv("../data/train_u6lujuX_CVtuZ9i.csv")
train_data.head()
```

[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coapplicant_Income	Loan_Amount	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

[ ]:
```
print(train_data.shape)
```

```
(614, 13)
```

[ ]:
```
train_data.describe()
```

[ ]:

	Applicant_Income	Coapplicant_Income	Loan_Amount	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

[ ]:
```
# variable types
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   Applicant_Income    614 non-null    int64
 7   Coapplicant_Income  614 non-null    float64
 8   Loan_Amount         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
```

[ ]:
```
# WORK WITH MISSING VALUES AND DROPPING UNNECESSARY COLUMNS
```

[ ]:
```
# count missing values per col
def missing_values(df):
    num_null_values = df.isnull().sum()
    return num_null_values
```

[ ]:
```
missing_values(train_data)
```

[ ]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
Applicant_Income      0
Coapplicant_Income    0
Loan_Amount          22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

[ ]:
```
# drop cols, mutate df in place
# Dependents col includes 3+ value, which makes processing complicated - numerical vs categorical
train_data.drop(["Loan_ID","Dependents"], axis=1, inplace=True)
```

[ ]:
```
train_data
```

[ ]:
```
     Gender Married     Education Self_Employed  Applicant_Income  \
0      Male      No      Graduate            No              5849
1      Male     Yes      Graduate            No              4583
2      Male     Yes      Graduate           Yes              3000
3      Male     Yes  Not Graduate            No              2583
4      Male      No      Graduate            No              6000
..      …     …           …            …                …
609  Female      No      Graduate            No              2900
610    Male     Yes      Graduate            No              4106
611    Male     Yes      Graduate            No              8072
612    Male     Yes      Graduate            No              7583
613  Female      No      Graduate           Yes              4583

     Coapplicant_Income  Loan_Amount  Loan_Amount_Term  Credit_History  \
0                   0.0          NaN             360.0             1.0
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
..                  …          …                 …                 …
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 Loan_Status
0           Urban           Y
1           Rural           N
2           Urban           Y
3           Urban           Y
4           Urban           Y
..            …         …
609         Rural           Y
610         Rural           Y
611         Urban           Y
612         Urban           Y
613     Semiurban           N

[614 rows x 11 columns]
```

[ ]:
```
# Dealing with null values (categorical)

cols = train_data[["Gender", "Married", "Self_Employed"]]
for i in cols:
    # fill missing values with mode of column, e.g. replace missing gender with␣
 ↪Male
    train_data[i].fillna(train_data[i].mode().iloc[0], inplace=True)
```

[ ]:
```
# no more missing values for categorical features
train_data.isnull().sum()
```

[ ]:
```
Gender                0
Married               0
Education             0
Self_Employed         0
Applicant_Income      0
Coapplicant_Income    0
Loan_Amount          22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

[ ]:
```
## Dealing with missing values (numerical) ##
n_cols = train_data[["Loan_Amount", "Loan_Amount_Term", "Credit_History"]]
for i in n_cols:
    # fill missing numerical values with mean of feature
    train_data[i].fillna(train_data[i].mean(axis=0), inplace=True)
```

[ ]:
```
# VISUALIZATION
```

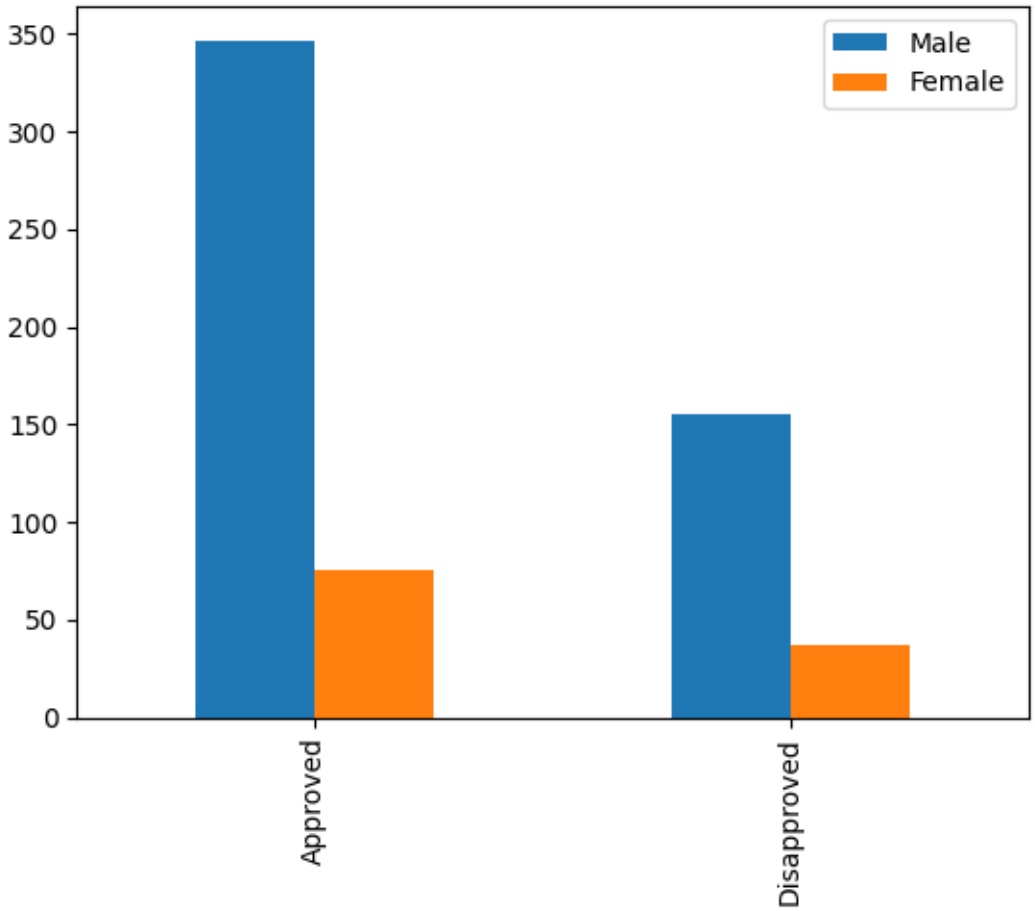
[ ]:
```
def bar_chart(col):
    Approved = train_data[train_data["Loan_Status"]=="Y"][col].value_counts()
    Disapproved = train_data[train_data["Loan_Status"]=="N"][col].value_counts()

    print(Approved)
    print(Disapproved)

    df1 = pd.DataFrame([Approved, Disapproved])
    df1.index = ["Approved", "Disapproved"]
    df1.plot(kind="bar")
```

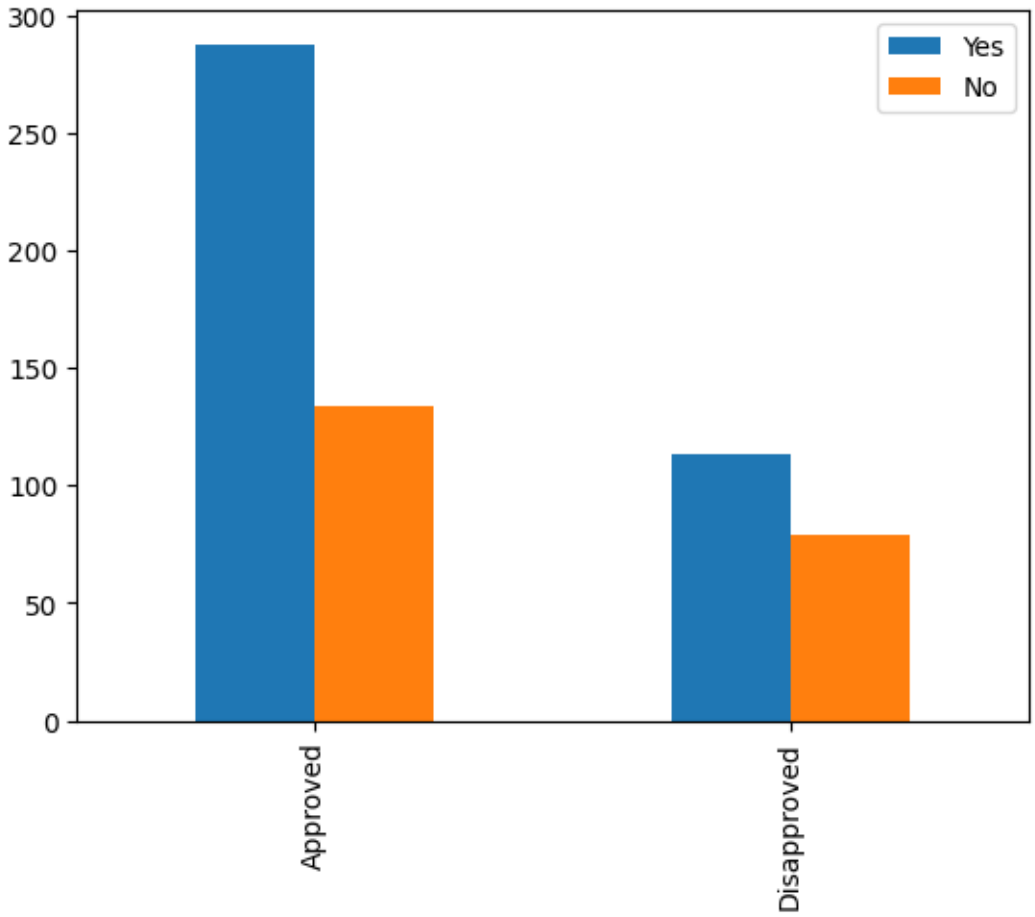
[ ]:
```
bar_chart("Gender")
```

```
Male      347
Female     75
Name: Gender, dtype: int64
Male      155
Female     37
Name: Gender, dtype: int64
```



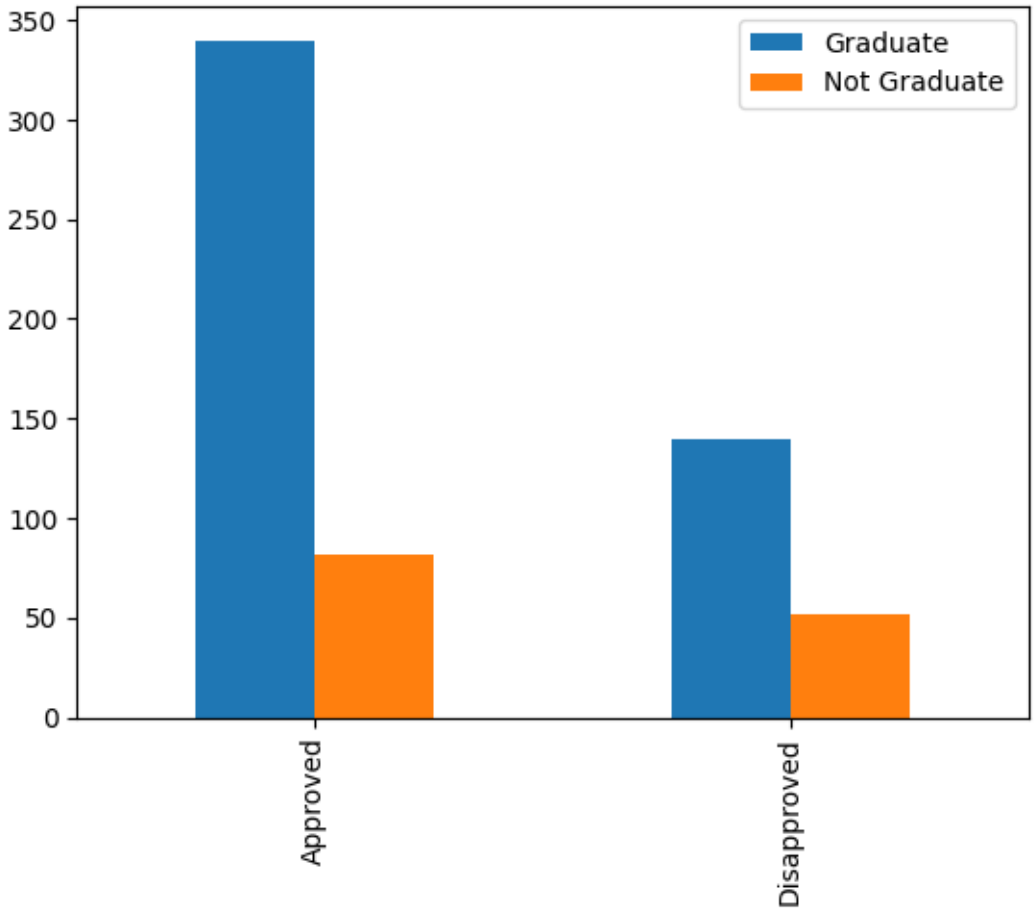
[ ]:
```
bar_chart("Married")
```

```
Yes    288
No     134
Name: Married, dtype: int64
Yes    113
No      79
Name: Married, dtype: int64
```



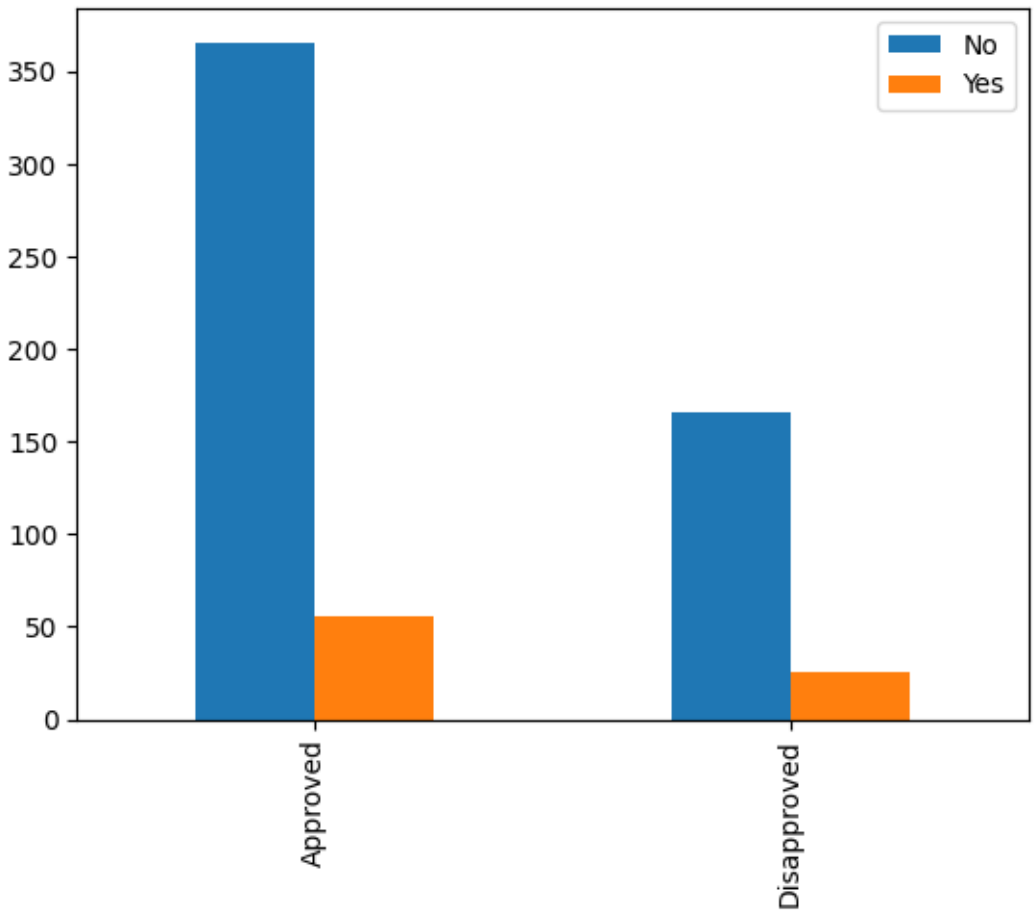
[ ]:
```
bar_chart("Education")
```

```
Graduate        340
Not Graduate     82
Name: Education, dtype: int64
Graduate        140
Not Graduate     52
Name: Education, dtype: int64
```



[ ]:
```
bar_chart("Self_Employed")
```

```
No     366
Yes     56
Name: Self_Employed, dtype: int64
No     166
Yes     26
Name: Self_Employed, dtype: int64
```



[ ]:
```
from sklearn.preprocessing import OrdinalEncoder


ord_enc = OrdinalEncoder()
# encode categories into numbers so machine can understand
train_data[["Gender",'Married','Education','Self_Employed','Property_Area','Loan_Status']]␣
 ↪= ord_enc.
 ↪fit_transform(train_data[["Gender",'Married','Education','Self_Employed','Property_Area','L
train_data.head()
```

[ ]:
```
   Gender  Married  Education  Self_Employed  Applicant_Income  \
0     1.0      0.0        0.0            0.0              5849
1     1.0      1.0        0.0            0.0              4583
2     1.0      1.0        0.0            1.0              3000
3     1.0      1.0        1.0            0.0              2583
4     1.0      0.0        0.0            0.0              6000

   Coapplicant_Income  Loan_Amount  Loan_Amount_Term  Credit_History  \
```

```
0                 0.0   146.412162             360.0             1.0
1              1508.0   128.000000             360.0             1.0
2                 0.0    66.000000             360.0             1.0
3              2358.0   120.000000             360.0             1.0
4                 0.0   141.000000             360.0             1.0

   Property_Area  Loan_Status
0            2.0          1.0
1            0.0          0.0
2            2.0          1.0
3            2.0          1.0
4            2.0          1.0
```

[ ]:
```
train_data[["Gender",'Married','Education','Self_Employed','Property_Area','Loan_Status']]␣
↪=␣
↪train_data[["Gender",'Married','Education','Self_Employed','Property_Area','Loan_Status']].
↪astype('int')
train_data.head()
```

[ ]:
```
   Gender  Married  Education  Self_Employed  Applicant_Income  \
0       1        0          0              0              5849
1       1        1          0              0              4583
2       1        1          0              1              3000
3       1        1          1              0              2583
4       1        0          0              0              6000

   Coapplicant_Income  Loan_Amount  Loan_Amount_Term  Credit_History  \
0                 0.0   146.412162             360.0             1.0
1              1508.0   128.000000             360.0             1.0
2                 0.0    66.000000             360.0             1.0
3              2358.0   120.000000             360.0             1.0
4                 0.0   141.000000             360.0             1.0

   Property_Area  Loan_Status
0              2            1
1              0            0
2              2            1
3              2            1
4              2            1
```

[ ]:
```
from sklearn.model_selection import train_test_split
# remove target variable from feature set
X = train_data.drop("Loan_Status", axis=1)
# store target variable in y
y = train_data["Loan_Status"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=True)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(491, 10)
(491,)
(123, 10)
(123,)
```

[ ]:
```
from sklearn.naive_bayes import GaussianNB
# data follows normal distribution

gfc = GaussianNB()
# learn what kind of features result in which kind of loan statuses
gfc.fit(X_train, y_train)
# run against X_test
pred1 = gfc.predict(X_test)
```

[ ]:
```
from sklearn.metrics import precision_score, recall_score, accuracy_score

# why use precision and recall? when dataset is imbalanced
def loss(y_true, y_pred):
    # how many correct TP / TP+FP
    pre = precision_score(y_true, y_pred)
    # how many correct TP / TP+FN
    rec = recall_score(y_true, y_pred)
    acc = accuracy_score(y_true, y_pred)

    print('precision', pre)
    print('recall', rec)
    print('accuracy', acc)
```

[ ]:
```
loss(y_test, pred1)
```

```
precision 0.7920792079207921
recall 0.9523809523809523
accuracy 0.7967479674796748
```

[ ]:
```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

# defining parameter range
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=0.687 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.694 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=0.687 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.694 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=0.687 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.694 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=0.687 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.694 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=0.687 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.694 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=0.687 total time=   0.0s
[CV 2/5] END …C=1, gamma=1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=1, gamma=1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=1, gamma=1, kernel=rbf;, score=0.694 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=0.697 total time=   0.0s
[CV 2/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 5/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.704 total time=   0.0s
[CV 1/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.697 total time=   0.0s
[CV 2/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.684 total time=   0.0s
[CV 5/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.704 total time=   0.0s
```

```
[CV 1/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.687 total time=   0.0s
[CV 2/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.673 total time=   0.0s
[CV 3/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.704 total time=   0.0s
[CV 4/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.663 total time=   0.0s
[CV 5/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.714 total time=   0.0s
[CV 1/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.646 total time=   0.0s
[CV 2/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.653 total time=   0.0s
[CV 3/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.633 total time=   0.0s
[CV 4/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.663 total time=   0.0s
[CV 5/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.673 total time=   0.0s
[CV 1/5] END …C=10, gamma=1, kernel=rbf;, score=0.697 total time=   0.0s
[CV 2/5] END …C=10, gamma=1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=10, gamma=1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=10, gamma=1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 5/5] END …C=10, gamma=1, kernel=rbf;, score=0.704 total time=   0.0s
[CV 1/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.697 total time=   0.0s
[CV 2/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 5/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.704 total time=   0.0s
[CV 1/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.697 total time=   0.0s
[CV 2/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.673 total time=   0.0s
[CV 5/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.704 total time=   0.0s
[CV 1/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.677 total time=   0.0s
[CV 2/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.653 total time=   0.0s
[CV 3/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.663 total time=   0.0s
[CV 5/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.714 total time=   0.0s
[CV 1/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.616 total time=   0.0s
[CV 2/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.622 total time=   0.0s
[CV 3/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.622 total time=   0.0s
[CV 4/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.622 total time=   0.0s
[CV 5/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.663 total time=   0.0s
[CV 1/5] END …C=100, gamma=1, kernel=rbf;, score=0.697 total time=   0.0s
[CV 2/5] END …C=100, gamma=1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=100, gamma=1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=100, gamma=1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 5/5] END …C=100, gamma=1, kernel=rbf;, score=0.704 total time=   0.0s
[CV 1/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.697 total time=   0.0s
[CV 2/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 5/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.704 total time=   0.0s
[CV 1/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.697 total time=   0.0s
[CV 2/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.684 total time=   0.0s
```

```
[CV 4/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.673 total time=   0.0s
[CV 5/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.704 total time=   0.0s
[CV 1/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.677 total time=   0.0s
[CV 2/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.653 total time=   0.0s
[CV 3/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.694 total time=   0.0s
[CV 4/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.673 total time=   0.0s
[CV 5/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.714 total time=   0.0s
[CV 1/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.606 total time=   0.0s
[CV 2/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.592 total time=   0.0s
[CV 3/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.582 total time=   0.0s
[CV 4/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.622 total time=   0.0s
[CV 5/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.673 total time=   0.0s
[CV 1/5] END …C=1000, gamma=1, kernel=rbf;, score=0.697 total time=   0.0s
[CV 2/5] END …C=1000, gamma=1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=1000, gamma=1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=1000, gamma=1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 5/5] END …C=1000, gamma=1, kernel=rbf;, score=0.704 total time=   0.0s
[CV 1/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.697 total time=   0.0s
[CV 2/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.694 total time=   0.0s
[CV 5/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.704 total time=   0.0s
[CV 1/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.697 total time=   0.0s
[CV 2/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.684 total time=   0.0s
[CV 3/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.684 total time=   0.0s
[CV 4/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.673 total time=   0.0s
[CV 5/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.704 total time=   0.0s
[CV 1/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.677 total time=   0.0s
[CV 2/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.653 total time=   0.0s
[CV 3/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.694 total time=   0.0s
[CV 4/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.673 total time=   0.0s
[CV 5/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.714 total time=   0.0s
[CV 1/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.606 total time=   0.0s
[CV 2/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.582 total time=   0.0s
[CV 3/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.592 total time=   0.0s
[CV 4/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.622 total time=   0.0s
[CV 5/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.673 total time=   0.0s
```

[ ]:
```
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)
```

[ ]:
```
grid.best_params_
```

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

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

```
precision 0.6829268292682927
recall 1.0
accuracy 0.6829268292682927
```

[ ]:
```
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)
```

```
precision 0.8152173913043478
recall 0.8928571428571429
accuracy 0.7886178861788617
```

[ ]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import RandomizedSearchCV

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

    # Extract best score
    best_score = rand_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
```

[ ]:
```
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':[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: 0.811
Test score: 0.805
```

[ ]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=2, max_features=0.95,
                       min_samples_leaf=0.04, min_samples_split=6,
                       random_state=2)
```

[ ]:
```
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)
```

```
precision 0.7830188679245284
recall 0.9880952380952381
accuracy 0.8048780487804879
```

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
```

```
randomized_search(params={
                          'min_samples_leaf':[1,2,4,6,8,10,20,30],
                           'min_impurity_decrease':[0.0, 0.01, 0.05, 0.10, 0.15,␣
 ↪0.2],
                           'max_features':[0.8, 0.7, 0.6, 0.5, 0.4],
                           'max_depth':[None,2,4,6,8,10,20],
                          }, clf=RandomForestClassifier(random_state=2))
```

```
Training score: 0.813
Test score: 0.805
```

[ ]:
```
RandomForestClassifier(max_depth=10, max_features=0.4, min_samples_leaf=6,
                       random_state=2)
```

[ ]:
```
import joblib
joblib.dump(ds, "model.pkl")
model = joblib.load('model.pkl' )
model_pred = model.predict(X_test.iloc[0].values.reshape(1,-1))
print(model_pred)

from datetime import datetime
print(f'completed (UTC) - {datetime.now()}')
```

```
[1]
completed (UTC) - 2022-12-17 16:13:00.096164
```

```
/home/codespace/.local/lib/python3.10/site-packages/sklearn/base.py:450:
UserWarning: X does not have valid feature names, but DecisionTreeClassifier was
fitted with feature names
  warnings.warn(
```