33323 - Aditya Kangune Assignment 1 LP Lab

Double-click (or enter) to edit

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
# from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_re
import random
```

Reading the CSV File

```
df = pd.read_csv("/content/drive/MyDrive/Datasets/heart_disease
```

```
df
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope
1	63	1	typical	145	233	1	2	150	0	2.3	3

Shape of dataset

4	37	1	nonanginal	130	250	0	0	187	0	3.5	3

```
df.shape
```

```
(303, 14)
```

299	45	1	typical	110	264	0	0	132	0	1.2	2

Count of columns

302	57	0	nontypical	130	236	0	2	174	0	0.0	2

```
df.count()
```

```
Age          303
Sex          303
ChestPain    303
RestBP       303
Chol         303
Fbs          303
RestECG      303
MaxHR        303
ExAng        303
Oldpeak      303
Slope        303
Ca           299
Thal         301
AHD          303
dtype: int64
```

Describing the dataset

```
df.describe()
```

	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxH
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0000

Displaying info about the dataset

min	29.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.0000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 303 entries, 1 to 303
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Age        303 non-null    int64  
 1   Sex        303 non-null    int64  
 2   ChestPain  303 non-null    object 
 3   RestBP     303 non-null    int64  
 4   Chol       303 non-null    int64  
 5   Fbs        303 non-null    int64  
 6   RestECG    303 non-null    int64  
 7   MaxHR      303 non-null    int64  
 8   ExAng      303 non-null    int64  
 9   Oldpeak    303 non-null    float64
 10  Slope      303 non-null    int64  
 11  Ca         299 non-null    float64
 12  Thal       301 non-null    object 
 13  AHD        303 non-null    object 
dtypes: float64(2), int64(9), object(3)
memory usage: 35.5+ KB
```

Displaying first 5 records

```
df.head()
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	
1	63	1	typical	145	233	1	2	150	0	2.3	3	(
2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3
3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2
4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	(
5	41	0	nontypical	130	204	0	2	172	0	1.4	1	(

Displaying last 5 records

```
df.tail()
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope
299	45	1	typical	110	264	0	0	132	0	1.2	2
300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2
301	57	1	asymptomatic	130	131	0	0	115	1	1.2	2
302	57	0	nontypical	130	236	0	2	174	0	0.0	2
303	38	1	nonanginal	138	175	0	0	173	0	0.0	1

Handling Categorical Data in Python

Categorical features are typically stored as text values which represent various traits of the observations. For example, gender is described as Male (M) or Female (F), product type could be described as electronics, apparels, food etc.

The boxplot is a simple way of representing statistical data on a plot in which a rectangle is drawn to represent the second and third quartiles, usually with a vertical line inside to indicate the median value.

```
for col in df.columns:
  if len(df[col].unique()) < 10:
    print(col, df[col].unique())
    print("----------------------------------------")
    print(col, df[col].value_counts(), sep="\n")
    print("----------------------------------------")
```

```
Sex [1 0]
----------------------------------
Sex
1    206
0     97
Name: Sex, dtype: int64
----------------------------------
ChestPain ['typical' 'asymptomatic' 'nonanginal' 'nontypical']
----------------------------------
ChestPain
asymptomatic    144
nonanginal       86
nontypical       50
typical          23
Name: ChestPain, dtype: int64
----------------------------------
Fbs [1 0]
----------------------------------
Fbs
```

```
0    258
1     45
Name: Fbs, dtype: int64
-------------------------------------
RestECG [2 0 1]
-------------------------------------
RestECG
0    151
2    148
1      4
Name: RestECG, dtype: int64
-------------------------------------
ExAng [0 1]
-------------------------------------
ExAng
0    204
1     99
Name: ExAng, dtype: int64
-------------------------------------
Slope [3 2 1]
-------------------------------------
Slope
1    142
2    140
3     21
Name: Slope, dtype: int64
-------------------------------------
Ca [ 0.  3.  2.  1. nan]
-------------------------------------
Ca
0.0    176
1.0     65
2.0     38
3.0     20
Name: Ca, dtype: int64
-------------------------------------
Thal ['fixed' 'normal' 'reversable' nan]
-------------------------------------
Thal
```

Displaying null values

```
df.isnull()
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slo
1	False	False	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	False	False	Fa
3	False	False	False	False	False	False	False	False	False	False	Fa
4	False	False	False	False	False	False	False	False	False	False	Fa
5	False	False	False	False	False	False	False	False	False	False	Fa

Getting count of null values per column

```
df.isnull().sum()
```

```
Age          0
Sex          0
ChestPain    0
RestBP       0
Chol         0
Fbs          0
RestECG      0
MaxHR        0
ExAng        0
Oldpeak      0
Slope        0
Ca           4
Thal         2
AHD          0
dtype: int64
```

Getting total count of null values

```
df.isnull().sum().sum()
```

```
6
```

Filling null values

```
df["Ca"].fillna( method ='ffill', inplace = True)
```

```
df["Thal"].fillna( method ='ffill', inplace = True)
```

```
df.isnull().sum()
```

```
Age          0
Sex          0
ChestPain    0
RestBP       0
Chol         0
Fbs          0
RestECG      0
MaxHR        0
ExAng        0
Oldpeak      0
Slope        0
Ca           0
Thal         0
AHD          0
dtype: int64
```

Sorting according to age

```
df.sort_values("Age")
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope
133	29	1	nontypical	130	204	0	2	202	0	0.0	1
102	34	1	typical	118	182	0	2	174	0	0.0	1
226	34	0	nontypical	118	210	0	0	192	0	0.7	1
284	35	1	nontypical	122	192	0	0	174	0	0.0	1
118	35	0	asymptomatic	138	183	0	0	182	0	1.4	1
...	...	...	...	...	...	...	...	...	...	...	...
43	71	0	nontypical	160	302	0	0	162	0	0.4	1
104	71	0	nonanginal	110	265	1	2	130	0	0.0	1
234	74	0	nontypical	120	269	0	2	121	1	0.2	1
258	76	0	nonanginal	140	197	0	1	116	0	1.1	2
162	77	1	asymptomatic	125	304	0	2	162	1	0.0	1

303 rows × 14 columns

Sorting in reverse order according to Age

```
df.sort_values("Age", ascending=False, kind="mergesort")
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope
162	77	1	asymptomatic	125	304	0	2	162	1	0.0	1
258	76	0	nonanginal	140	197	0	1	116	0	1.1	2
234	74	0	nontypical	120	269	0	2	121	1	0.2	1
43	71	0	nontypical	160	302	0	0	162	0	0.4	1
104	71	0	nonanginal	110	265	1	2	130	0	0.0	1
...	...	...	...	...	...	...	...	...	...	...	...
169	35	1	asymptomatic	126	282	0	2	156	1	0.0	1
284	35	1	nontypical	122	192	0	0	174	0	0.0	1
102	34	1	typical	118	182	0	2	174	0	0.0	1
226	34	0	nontypical	118	210	0	0	192	0	0.7	1
133	29	1	nontypical	130	204	0	2	202	0	0.0	1

Sorting by Multiple Columns in Ascending Order

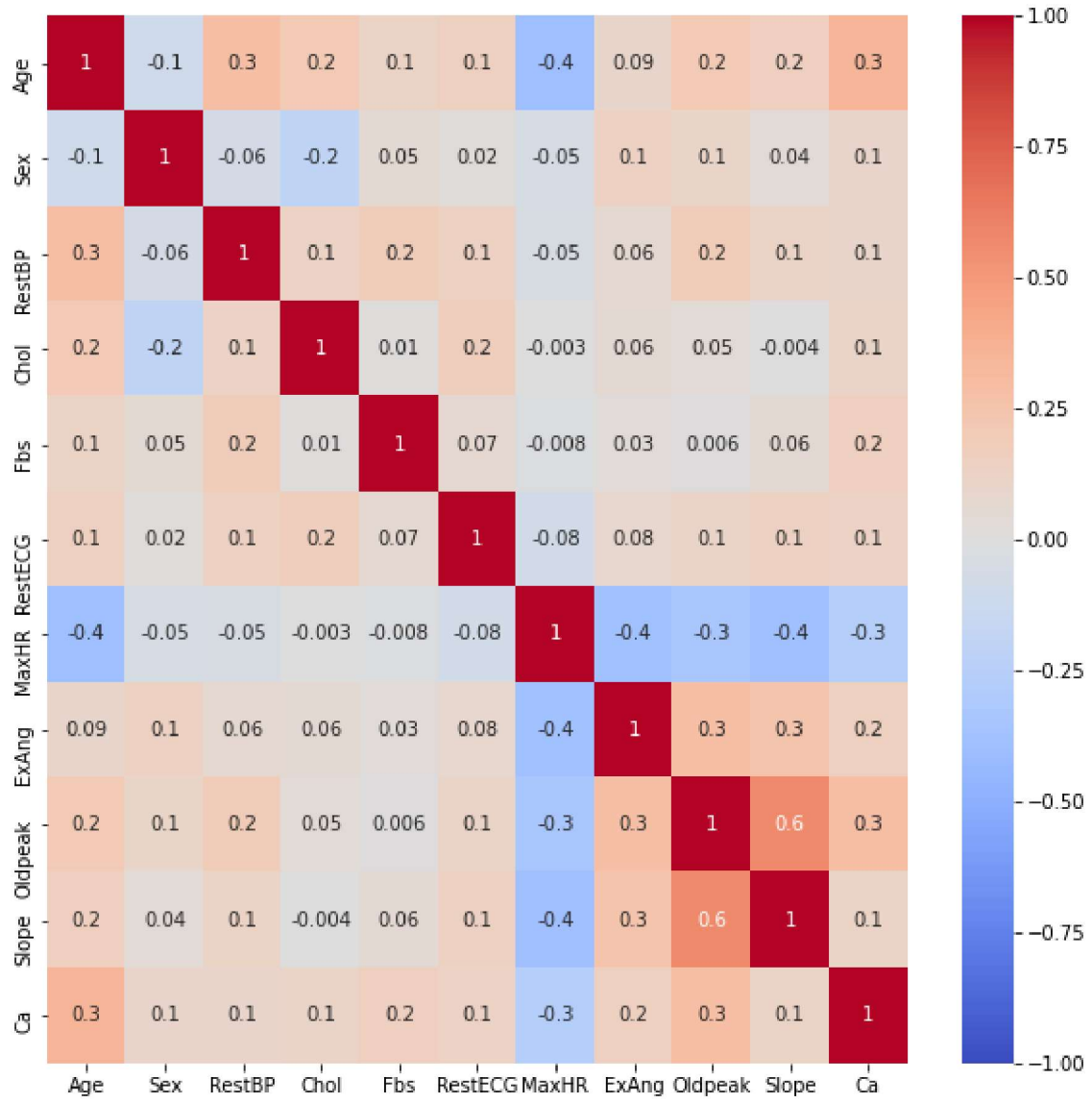
```
df.sort_values(by=["Age", "RestBP"])
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope
133	29	1	nontypical	130	204	0	2	202	0	0.0	1
102	34	1	typical	118	182	0	2	174	0	0.0	1
226	34	0	nontypical	118	210	0	0	192	0	0.7	1
139	35	1	asymptomatic	120	198	0	0	130	1	1.6	2
284	35	1	nontypical	122	192	0	0	174	0	0.0	1
...	...	...	...	...	...	...	...	...	...	...	...
274	71	0	asymptomatic	112	149	0	0	125	0	1.6	2
43	71	0	nontypical	160	302	0	0	162	0	0.4	1
234	74	0	nontypical	120	269	0	2	121	1	0.2	1
258	76	0	nonanginal	140	197	0	1	116	0	1.1	2
162	77	1	asymptomatic	125	304	0	2	162	1	0.0	1

303 rows × 14 columns

Heatmap

```
plt.figure(figsize=(10, 10))
sns.heatmap(df.corr(), annot=True, center=0, vmin=-1, vmax=1, f
plt.show()
```




```
# sns.heatmap(df["ChestPain"])
# plt.show()

# a = df.iloc[:, [1, 3]]
# print(a)

X = df.iloc[:, :-1].values
# print(X)

y = df.iloc[:, -1].values
print(y)
```

```
['No' 'Yes' 'Yes' 'No' 'No' 'No' 'Yes' 'No' 'Yes' 'Yes' 'No' 'No' 'Yes'
```

```
'No' 'No' 'No' 'Yes' 'No' 'No' 'No' 'No' 'No' 'Yes' 'Yes' 'Yes' 'No' 'No'
'No' 'No' 'Yes' 'No' 'Yes' 'Yes' 'No' 'No' 'No' 'Yes' 'Yes' 'Yes' 'No'
'Yes' 'No' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'No' 'No' 'No' 'Yes'
'No' 'Yes' 'Yes' 'Yes' 'Yes' 'No' 'No' 'Yes' 'No' 'Yes' 'No' 'Yes' 'Yes'
'Yes' 'No' 'Yes' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'No' 'No'
'Yes' 'No' 'No' 'No' 'Yes' 'No' 'No' 'No' 'No' 'No' 'No' 'No' 'Yes' 'No'
'No' 'No' 'Yes' 'Yes' 'Yes' 'No' 'No' 'No' 'No' 'No' 'No' 'Yes' 'No'
'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'Yes' 'No' 'No' 'No' 'Yes'
'Yes' 'Yes' 'Yes' 'No' 'Yes' 'Yes' 'No' 'Yes' 'Yes' 'No' 'No' 'No' 'No'
'No' 'No' 'No' 'No' 'Yes' 'Yes' 'Yes' 'No' 'No' 'Yes' 'No' 'Yes' 'No'
'Yes' 'Yes' 'No' 'No' 'No' 'No' 'No' 'No' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes'
'Yes' 'No' 'No' 'Yes' 'No' 'No' 'No' 'No' 'No' 'No' 'Yes' 'No' 'Yes' 'No'
'Yes' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'No' 'Yes'
'No' 'No' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'No' 'No'
'No' 'Yes' 'No' 'No' 'No' 'No' 'No' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'No'
'Yes' 'No' 'Yes' 'Yes' 'No' 'No' 'No' 'No' 'No' 'No' 'No' 'No' 'Yes'
'Yes' 'No' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'Yes' 'No' 'No' 'Yes' 'Yes'
'Yes' 'No' 'No' 'No' 'No' 'No' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'Yes' 'No'
'No' 'Yes' 'No' 'No' 'No' 'No' 'No' 'No' 'No' 'Yes' 'No' 'Yes' 'No' 'No'
'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes' 'No' 'Yes' 'No' 'No'
'No' 'Yes' 'No' 'Yes' 'No' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'No' 'No' 'No'
'Yes' 'No' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes'
'No']
```

```
from sklearn.model_selection import train_test_split
# X: Independant variable, y : Dependant variable.
# Parameters for train_test_split:
            # X - Matrix features
            # y - Dependant variable vector
            # Test size - 20% recommended
            # random_state - So that we get the same randomized
X_train, X_test, y_train, y_test = train_test_split(X, y, test_
```

```
X_train
```

```
array([[58, 1, 'asymptomatic', ..., 2, 3.0, 'reversable'],
       [54, 1, 'asymptomatic', ..., 2, 1.0, 'reversable'],
       [56, 1, 'asymptomatic', ..., 2, 1.0, 'normal'],
       ...,
       [62, 1, 'asymptomatic', ..., 2, 2.0, 'reversable'],
       [54, 1, 'asymptomatic', ..., 2, 2.0, 'normal'],
       [57, 1, 'asymptomatic', ..., 2, 1.0, 'fixed']], dtype=object)
```

```
X_test
```

```
array([[43, 1, 'asymptomatic', 110, 211, 0, 0, 161, 0, 0.0, 1, 0.0,
        'reversable'],
       [68, 1, 'nonanginal', 118, 277, 0, 0, 151, 0, 1.0, 1, 1.0,
        'reversable'],
       [59, 1, 'asymptomatic', 138, 271, 0, 2, 182, 0, 0.0, 1, 0.0,
        'normal'],
       [64, 1, 'asymptomatic', 145, 212, 0, 2, 132, 0, 2.0, 2, 2.0,
        'fixed'],
```

```
       [60, 0, 'asymptomatic', 158, 305, 0, 2, 161, 0, 0.0, 1, 0.0,
        'normal'],
       [41, 1, 'nontypical', 120, 157, 0, 0, 182, 0, 0.0, 1, 0.0,
        'normal'],
       [42, 1, 'nonanginal', 130, 180, 0, 0, 150, 0, 0.0, 1, 0.0,
        'normal'],
       [65, 1, 'asymptomatic', 135, 254, 0, 2, 127, 0, 2.8, 2, 1.0,
        'reversable'],
       [62, 0, 'asymptomatic', 138, 294, 1, 0, 106, 0, 1.9, 2, 3.0,
        'normal'],
       [64, 1, 'asymptomatic', 120, 246, 0, 2, 96, 1, 2.2, 3, 1.0,
        'normal'],
       [65, 1, 'asymptomatic', 120, 177, 0, 0, 140, 0, 0.4, 1, 0.0,
        'reversable'],
       [62, 1, 'nontypical', 128, 208, 1, 2, 140, 0, 0.0, 1, 0.0,
        'normal'],
       [57, 1, 'nontypical', 154, 232, 0, 2, 164, 0, 0.0, 1, 1.0,
        'normal'],
       [66, 1, 'asymptomatic', 112, 212, 0, 2, 132, 1, 0.1, 1, 1.0,
        'normal'],
       [55, 1, 'nontypical', 130, 262, 0, 0, 155, 0, 0.0, 1, 0.0,
        'normal'],
       [38, 1, 'nonanginal', 138, 175, 0, 0, 173, 0, 0.0, 1, 1.0,
        'normal'],
       [54, 1, 'nonanginal', 125, 273, 0, 2, 152, 0, 0.5, 3, 1.0,
        'normal'],
       [41, 0, 'nonanginal', 112, 268, 0, 2, 172, 1, 0.0, 1, 0.0,
        'normal'],
       [48, 0, 'nonanginal', 130, 275, 0, 0, 139, 0, 0.2, 1, 0.0,
        'normal'],
       [54, 1, 'asymptomatic', 110, 206, 0, 2, 108, 1, 0.0, 2, 1.0,
        'normal'],
       [56, 0, 'nontypical', 140, 294, 0, 2, 153, 0, 1.3, 2, 0.0,
        'normal'],
       [57, 1, 'asymptomatic', 130, 131, 0, 0, 115, 1, 1.2, 2, 1.0,
        'reversable'],
       [65, 0, 'nonanginal', 155, 269, 0, 0, 148, 0, 0.8, 1, 0.0,
        'normal'],
       [57, 1, 'asymptomatic', 165, 289, 1, 2, 124, 0, 1.0, 2, 3.0,
        'reversable'],
       [51, 1, 'nonanginal', 100, 222, 0, 0, 143, 1, 1.2, 2, 0.0,
        'normal'],
       [48, 1, 'nonanginal', 124, 255, 1, 0, 175, 0, 0.0, 1, 2.0,
        'normal'],
       [51, 1, 'asymptomatic', 140, 298, 0, 0, 122, 1, 4.2, 2, 3.0,
        'reversable'],
       [45, 1, 'asymptomatic', 104, 208, 0, 2, 148, 1, 3.0, 2, 0.0,
        'normal'],
       [66, 0, 'typical', 150, 226, 0, 0, 114, 0, 2.6, 3, 0.0, 'normal'],
       [55, 1, 'asymptomatic', 140, 217, 0, 0, 111, 1, 5.6, 3, 0.0,
```

y_train

```
array(['Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No',
       'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'No',
       'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'No', 'Yes',
       'No', 'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'Yes',
       'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'Yes',
       'Yes', 'No', 'No', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'No',
```

```
       'Yes', 'Yes', 'Yes', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No',
       'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes',
       'No', 'Yes', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No',
       'No', 'Yes', 'No', 'No', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No',
       'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes',
       'Yes', 'No', 'Yes', 'Yes', 'No', 'No', 'No', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'No', 'No', 'No',
       'Yes', 'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'Yes', 'Yes',
       'Yes', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'No',
       'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes',
       'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'No', 'No', 'No',
       'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No',
       'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'No',
       'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'No', 'Yes',
       'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No',
       'Yes', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'No',
       'No', 'No', 'Yes', 'Yes', 'No', 'No', 'No', 'No', 'No', 'Yes',
       'Yes', 'Yes'], dtype=object)
```

```
y_test
```

```
array(['No', 'No', 'No', 'Yes', 'Yes', 'No', 'No', 'Yes', 'Yes', 'Yes',
       'No', 'No', 'Yes', 'Yes', 'No', 'No', 'No', 'No', 'No', 'Yes',
       'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'No', 'Yes',
       'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes', 'No', 'No', 'Yes', 'Yes',
       'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'No', 'No', 'No', 'No',
       'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No',
       'Yes'], dtype=object)
```

Confusion Matrix

```
ones = np.ones(50, dtype=int)
print(ones)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```
zeros = np.zeros(50, dtype=int)
print(zeros)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

```
y_actual = np.concatenate((ones, zeros))
print(y_actual)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

```
y_actual
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
# onesPredict = np.ones(45, dtype=int)
# print(ones)
```

```
# zerosPredict = np.zeros(55, dtype=int)
# print(zeros)
```

```
# y_pred = np.concatenate((ones, zeros))
# print(y_pred)
```

```
y_actual = np.array([0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0])
```

```
y_pred = np.array([0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1])
```

```
# random.shuffle(y_actual)
```

```
# random.shuffle(y_pred)
```

```
print(y_actual)
```

```
[0 1 1 1 1 0 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 0 1 0 0
 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1
 1 1 1 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 0]
```

```
print(y_pred)
```

```
[0 1 1 1 1 0 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 0 1 0 0
 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1
 1 1 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1]
```

```
classification_report(y_actual, y_pred)
```

```
'              precision    recall  f1-score   support\n\n           0       0.85
47\n           1       0.98      0.85      0.91        53\n\n    accuracy
91       100\n   macro avg       0.92      0.91      0.91       100\nweighted avg
```

```
con_mat = confusion_matrix(y_actual, y_pred)
print(con_mat)
```

```
[[46  1]
 [ 8 45]]
```

```
tn, fp, fn, tp = con_mat[0][0], con_mat[0][1], con_mat[1][0], c
print("True Negatives: ",tn)
print("False Positives: ",fp)
print("False Negatives: ",fn)
print("True Positives: ",tp)
```

```
True Negatives:  46
False Positives:  1
False Negatives:  8
True Positives:  45
```

Accuracy

```
Accuracy = (tn+tp)*100/(tp+tn+fp+fn)
print("Accuracy: {:0.2f}%".format(Accuracy))
```

```
Accuracy: 91.00%
```

Precision

```
Precision = tp/(tp+fp)
print("Precision: {:0.2f}".format(Precision))
```

```
Precision: 0.98
```

Recall

```
Recall = tp/(tp+fn)
print("Recall: {:0.2f}".format(Recall))
```

```
Recall: 0.85
```

F1 Score

```
f1 = (2*Precision*Recall)/(Precision + Recall)
print("F1 Score: {:0.2f}".format(f1))
```

```
F1 Score: 0.91
```

F-beta score calculation

```
def fbeta(precision, recall, beta):
    return ((1+pow(beta,2))*precision*recall)/(pow(beta,2)*prec

f2 = fbeta(Precision, Recall, 2)
f0_5 = fbeta(Precision, Recall, 0.5)

print("F2 {:0.2f}".format(f2))
print("\nF0.5 {:0.2f}".format(f0_5))
```

```
F2 0.87

F0.5 0.95
```

Specificity

```
Specificity = tn/(tn+fp)
print("Specificity: {:0.2f}".format(Specificity))
```

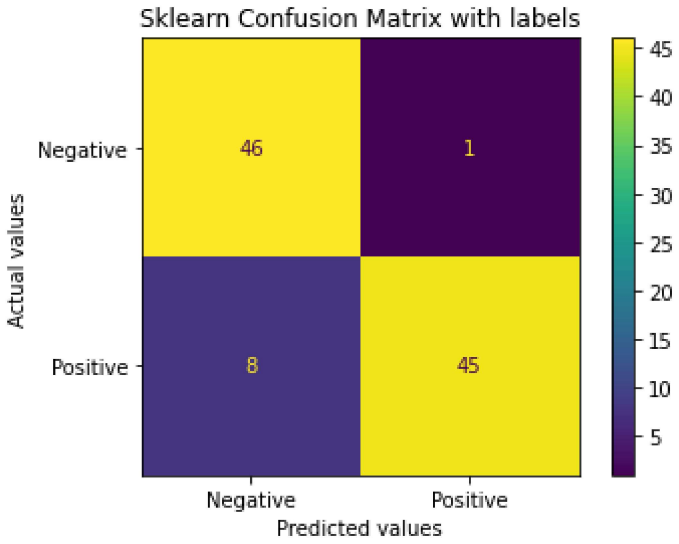
```
Specificity: 0.98
```

```
cmd_obj = ConfusionMatrixDisplay(con_mat, display_labels=["Nega

cmd_obj.plot()
```

```
cmd_obj.ax_.set(
                title='Sklearn Confusion Matrix with labels',
                xlabel='Predicted values',
                ylabel='Actual values')

plt.show()
```



Accuracy

```
91.0
```