```
from google.colab import files

uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving heart_disease.xlsx to heart_disease.xlsx
```

```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
from sklearn import tree

Heart_disease =
pd.read_excel("heart_disease.xlsx",sheet_name='Heart_disease')
Heart_disease
```


```
{"summary":"{\n  \"name\": \"Heart_disease\",\n  \"rows\": 908,\n
\"fields\": [\n    {\n      \"column\": \"age\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
9,\n        \"min\": 29,\n        \"max\": 77,\n
\"num_unique_values\": 49,\n        \"samples\": [\n          76,\n
73,\n          32\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sex\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"Female\",\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"cp\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          \"atypical angina\",\n          \"non-
anginal\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"trestbps\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 20,\n        \"min\": 0,\n
\"max\": 200,\n        \"num_unique_values\": 85,\n
\"samples\": [\n          186,\n          145\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 112,\n        \"min\": 0,\n
\"max\": 603,\n        \"num_unique_values\": 228,\n
\"samples\": [\n          176,\n          342\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fbs\",\n      \"properties\": {\n
\"dtype\": \"boolean\",\n        \"num_unique_values\": 2,\n
```

```
\"samples\": [\n          false,\n          true\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"restecg\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
3,\n        \"samples\": [\n          \"lv hypertrophy\",\n
\"normal\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"thalch\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 26,\n        \"min\": 60,\n        \"max\": 202,\n
\"num_unique_values\": 122,\n        \"samples\": [\n          131,\n
159\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"exang\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 4,\n        \"samples\": [\n
true,\n          \"TURE\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"oldpeak\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.0938749708784534,\n        \"min\": -
2.6,\n        \"max\": 6.2,\n        \"num_unique_values\": 53,\n
\"samples\": [\n          1.8,\n          5.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"slope\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"downsloping\",\n          \"flat\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"thal\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
3,\n        \"samples\": [\n          \"fixed defect\",\n
\"normal\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"num\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 0,\n        \"max\": 4,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          1,\n
4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"Heart_disease"}
```


```
Heart_disease.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 908 entries, 0 to 907
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       908 non-null    int64
 1   sex       908 non-null    object
 2   cp        908 non-null    object
 3   trestbps  908 non-null    int64
 4   chol      908 non-null    int64
 5   fbs       908 non-null    bool
```

```
 6   restecg   908 non-null    object
 7   thalch    908 non-null    int64
 8   exang     908 non-null    object
 9   oldpeak   846 non-null    float64
 10  slope     908 non-null    object
 11  thal      908 non-null    object
 12  num       908 non-null    int64
dtypes: bool(1), float64(1), int64(5), object(6)
memory usage: 86.1+ KB
```

```
Heart_disease.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
'thalch',
       'exang', 'oldpeak', 'slope', 'thal', 'num'],
      dtype='object')
```

```
print(Heart_disease.describe())
```

```
               age    trestbps        chol      thalch     oldpeak
num
count  908.000000  908.000000  908.000000  908.000000  846.000000
908.000000
mean    53.791850  133.430617  201.484581  135.957048    0.891253
1.008811
std      9.158031   20.401608  112.097949   26.804929    1.093875
1.144436
min     29.000000    0.000000    0.000000   60.000000   -2.600000
0.000000
25%     47.750000  120.000000  176.750000  118.000000    0.000000
0.000000
50%     54.000000  130.000000  224.000000  138.000000    0.500000
1.000000
75%     60.000000  144.000000  270.000000  156.000000    1.500000
2.000000
max     77.000000  200.000000  603.000000  202.000000    6.200000
4.000000
```

```
Heart_disease["num"].value_counts()
```

```
num
0    399
1    265
2    109
3    107
4     28
Name: count, dtype: int64
```

```
Heart_disease.isna().sum()
```

```
age          0
sex          0
cp           0
trestbps     0
chol         0
fbs          0
restecg      0
thalch       0
exang        0
oldpeak     62
slope        0
thal         0
num          0
dtype: int64
```

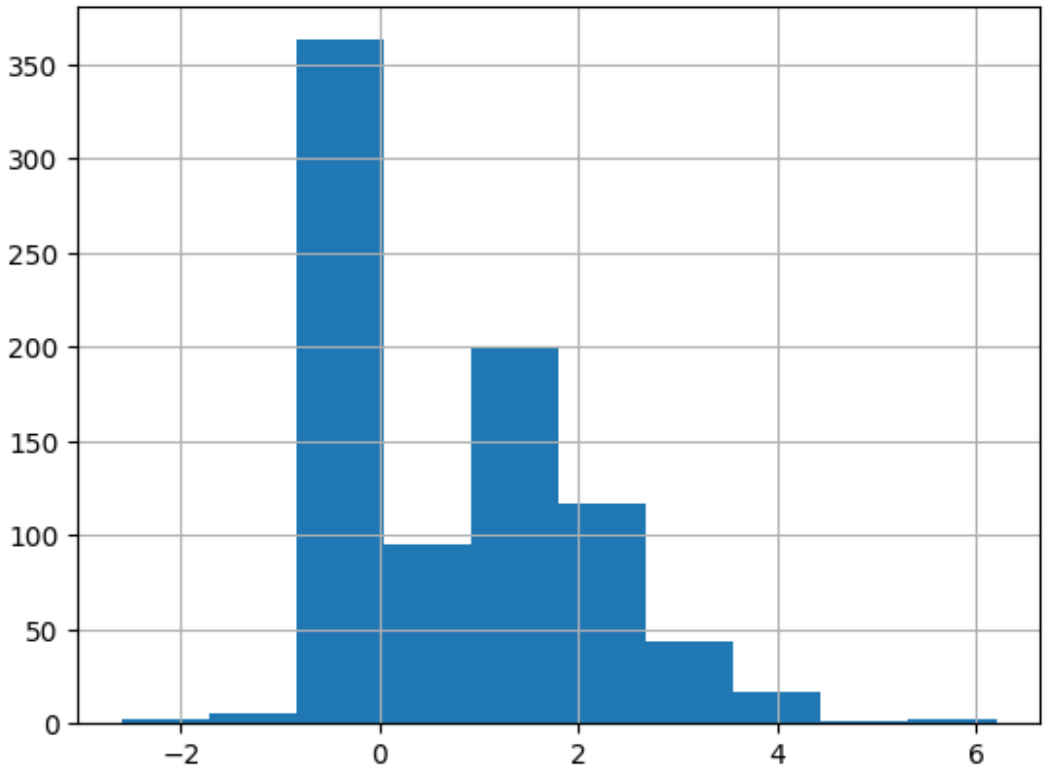
```
df = Heart_disease.dropna(axis=0)
df
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 846,\n  \"fields\": [\
n    {\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 9,\n        \"min\": 29,\n
\"max\": 77,\n        \"num_unique_values\": 49,\n        \"samples\":
[\n          76,\n          73,\n          32\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"sex\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"Female\",\n          \"Male\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"cp\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 4,\n
\"samples\": [\n          \"atypical angina\",\n          \"non-
anginal\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"trestbps\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 19,\n        \"min\": 0,\n
\"max\": 200,\n        \"num_unique_values\": 63,\n
\"samples\": [\n          0,\n          116\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 111,\n        \"min\": 0,\n
\"max\": 603,\n        \"num_unique_values\": 219,\n
\"samples\": [\n          287,\n          216\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fbs\",\n      \"properties\": {\n
\"dtype\": \"boolean\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          false,\n          true\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"restecg\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
3,\n        \"samples\": [\n          \"lv hypertrophy\",\n
```

\"normal\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"thalch\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 25,\n   \"min\": 60,\n   \"max\": 202,\n
\"num_unique_values\": 118,\n   \"samples\": [\n   188,\n
176\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"exang\",\n   \"properties\": {\n   \"dtype\": \"category\",\
n   \"num_unique_values\": 3,\n   \"samples\": [\n
false,\n   true\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"oldpeak\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 1.0938749708784534,\n   \"min\": -2.6,\n
\"max\": 6.2,\n   \"num_unique_values\": 53,\n
\"samples\": [\n   1.8,\n   5.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"slope\",\n   \"properties\": {\
n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n
\"samples\": [\n   \"downsloping\",\n   \"flat\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"thal\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
3,\n   \"samples\": [\n   \"fixed defect\",\n
\"normal\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"num\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1,\n   \"min\": 0,\n   \"max\": 4,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   1,\n
4\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

```
df["oldpeak"].hist()
```

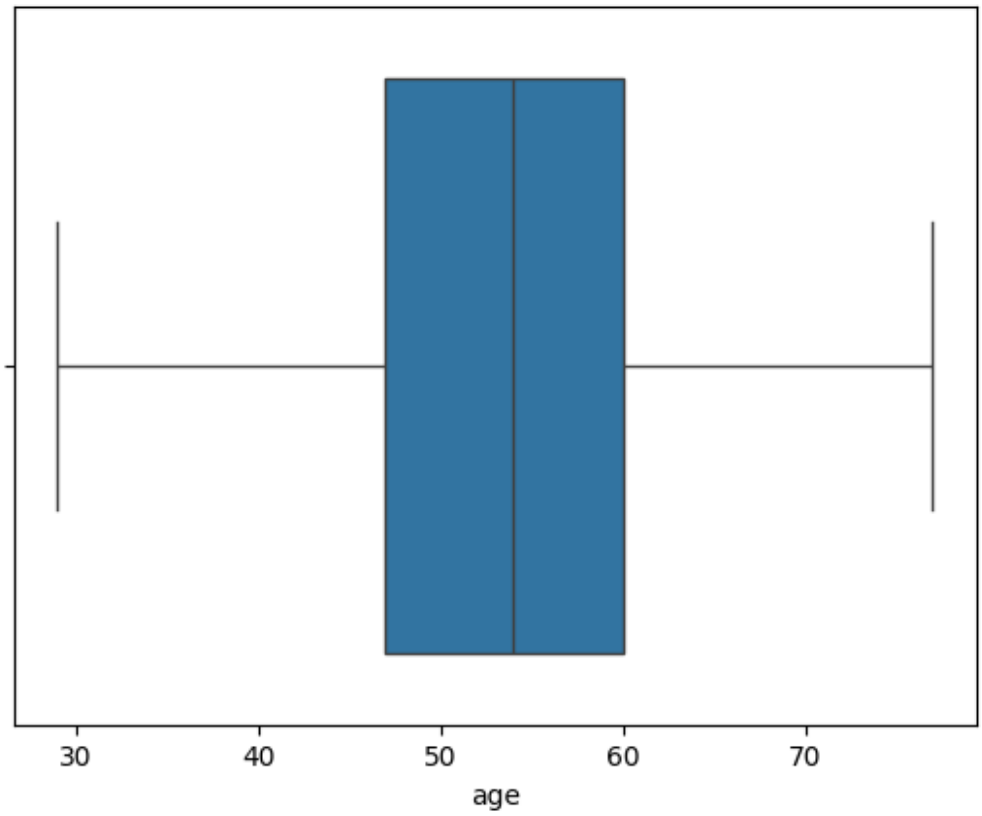
```
<Axes: >
```




```
df.shape
```

```
(846, 13)
```

```
sns.boxplot(x=df['age'])
plt.show()
```




```
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 846,\n  \"fields\": [\n    {\n      \"column\": \"age\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 9,\n        \"min\": 29,\n        \"max\": 77,\n        \"num_unique_values\": 49,\n        \"samples\": [\n          76,\n          73,\n          32\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"sex\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n        \"samples\": [\n          \"Female\",\n          \"Male\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"cp\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          \"atypical angina\",\n          \"non-anginal\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"trestbps\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 19,\n        \"min\": 0,\n        \"max\": 200,\n        \"num_unique_values\": 63,\n        \"samples\": [\n          0,\n          116\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 111,\n        \"min\": 0,\n
```

\"max\": 603,\n   \"num_unique_values\": 219,\n
\"samples\": [\n   287,\n   216\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"fbs\",\n   \"properties\": {\n
\"dtype\": \"boolean\",\n   \"num_unique_values\": 2,\n
\"samples\": [\n   false,\n   true\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"restecg\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
3,\n   \"samples\": [\n   \"lv hypertrophy\",\n
\"normal\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"thalch\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 25,\n   \"min\": 60,\n   \"max\": 202,\n
\"num_unique_values\": 118,\n   \"samples\": [\n   188,\n
176\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"exang\",\n   \"properties\": {\n   \"dtype\": \"category\",\
n   \"num_unique_values\": 3,\n   \"samples\": [\n
false,\n   true\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"oldpeak\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 1.0938749708784534,\n   \"min\": -2.6,\n
\"max\": 6.2,\n   \"num_unique_values\": 53,\n
\"samples\": [\n   1.8,\n   5.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"slope\",\n   \"properties\": {\
n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n
\"samples\": [\n   \"downsloping\",\n   \"flat\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"thal\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
3,\n   \"samples\": [\n   \"fixed defect\",\n
\"normal\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"num\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1,\n   \"min\": 0,\n   \"max\": 4,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   1,\n
4\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 846 entries, 0 to 905
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       846 non-null    int64
```

```
 1   sex       846 non-null    object
 2   cp        846 non-null    object
 3   trestbps  846 non-null    int64
 4   chol      846 non-null    int64
 5   fbs       846 non-null    bool
 6   restecg   846 non-null    object
 7   thalch    846 non-null    int64
 8   exang     846 non-null    object
 9   oldpeak   846 non-null    float64
 10  slope     846 non-null    object
 11  thal      846 non-null    object
 12  num       846 non-null    int64
dtypes: bool(1), float64(1), int64(5), object(6)
memory usage: 86.7+ KB
```

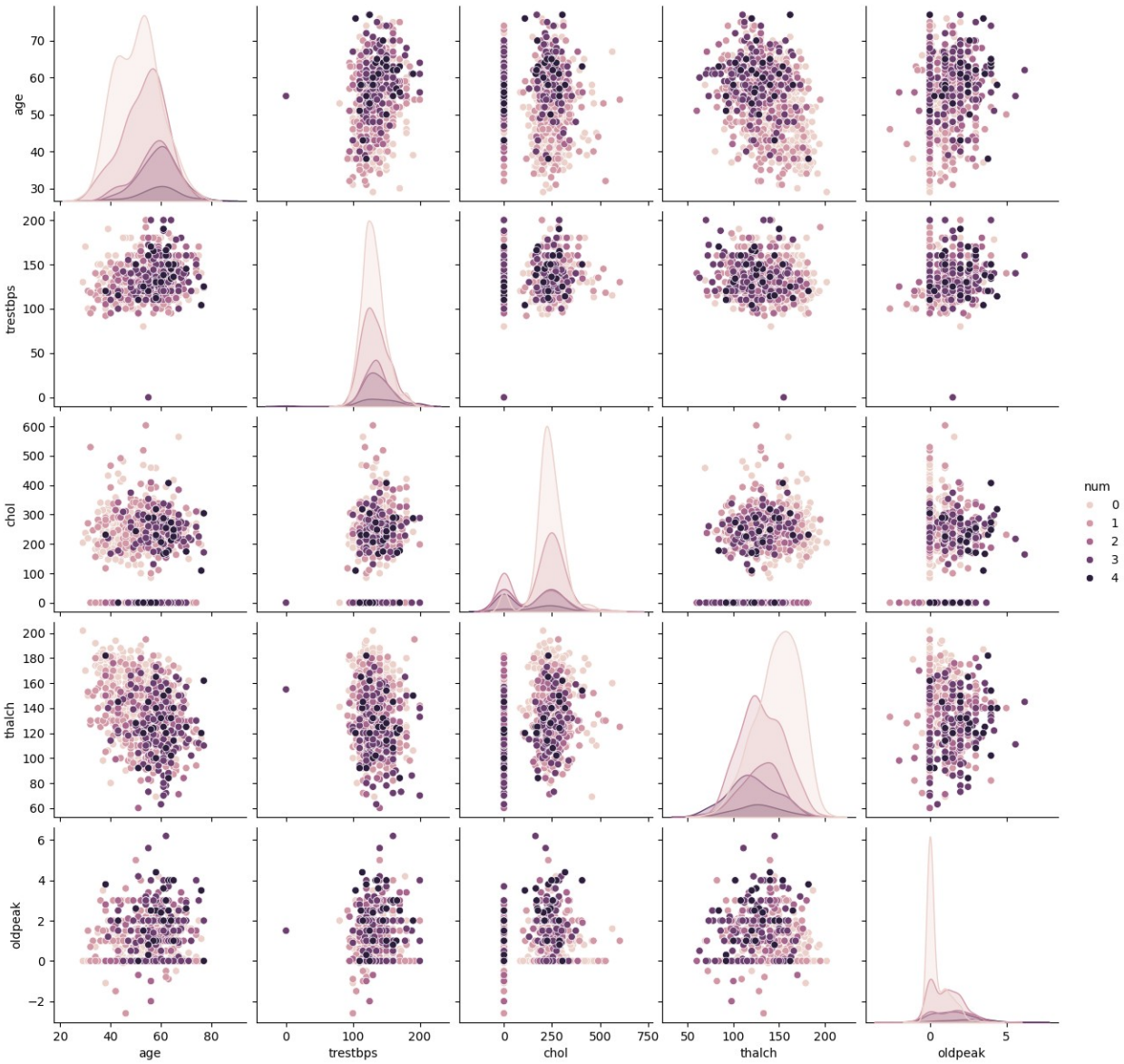
```
df[["age","trestbps","chol","thalch","oldpeak",'num']].corr()
```


```
{"summary":"{\n  \"name\":
\"df[[\\\"age\\\",\\\"trestbps\\\",\\\"chol\\\",\\\"thalch\\\",\\\"old
peak\\\",'num']]\",\n  \"rows\": 6,\n  \"fields\": [\n    {\n
\"column\": \"age\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.4580666778476807,\n        \"min\": -
0.34596007040006777,\n        \"max\": 1.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          1.0,\n
0.2443014447922987,\n          0.33139934040170166\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"trestbps\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.3836622996549028,\n        \"min\": -0.10939004693798397,\n
\"max\": 1.0,\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          0.2443014447922987,\n          1.0,\n
0.12615809725299584\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"chol\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.44153413085386645,\n        \"min\": -0.25619148332117864,\
n        \"max\": 1.0,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          -0.10747081283692353,\n
0.10091182403002281,\n          -0.25619148332117864\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"thalch\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.512106325223764,\n        \"min\": -0.3503310440976595,\n
\"max\": 1.0,\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          -0.34596007040006777,\n          -0.10939004693798397,\n
-0.3503310440976595\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"oldpeak\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 0.3988541761668519,\n        \"min\": -
0.13959814926378245,\n        \"max\": 1.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n
```

0.2426623961523708,\n   0.1671309784386287,\n
0.43757709343249257\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"num\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.4957474225725251,\n   \"min\": -0.3503310440976595,\n
\"max\": 1.0,\n   \"num_unique_values\": 6,\n   \"samples\":
[\n   0.33139934040170166,\n   0.12615809725299584,\n
1.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10,6))
sns.heatmap(df[["age","trestbps","chol","thalch","oldpeak",'num']].cor
r(),annot=True)
plt.show()
sns.pairplot(data=df[["age","trestbps","chol","thalch","oldpeak",'num'
]], hue = 'num')
```


```
<seaborn.axisgrid.PairGrid at 0x7ef6423f2710>
```




```
df["sex"].value_counts()

sex
Male      657
Female    189
Name: count, dtype: int64

df["cp"].value_counts()

cp
asymptomatic       464
non-anginal        184
atypical angina    158
typical angina      40
Name: count, dtype: int64
```

```
df["fbs"].value_counts()
```

```
fbs
False    688
True     158
Name: count, dtype: int64
```

```
df["restecg"].value_counts()
```

```
restecg
normal              520
lv hypertrophy      180
st-t abnormality    146
Name: count, dtype: int64
```

```
df["exang"].value_counts()
```

```
exang
False    512
True     333
FALSE      1
Name: count, dtype: int64
```

```
df.shape
```

```
(846, 13)
```

```
df["exang"].value_counts()
```

```
exang
False    512
True     333
FALSE      1
Name: count, dtype: int64
```

```
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 846,\n  \"fields\": [\n    {\n      \"column\": \"age\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 9,\n        \"min\": 29,\n        \"max\": 77,\n        \"num_unique_values\": 49,\n        \"samples\": [\n          76,\n          73,\n          32\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"sex\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n        \"samples\": [\n          \"Female\",\n          \"Male\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"cp\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          \"atypical angina\",\n          \"non-anginal\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

\"trestbps\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 19,\n   \"min\": 0,\n
\"max\": 200,\n   \"num_unique_values\": 63,\n
\"samples\": [\n   0,\n   116\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"chol\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 111,\n   \"min\": 0,\n
\"max\": 603,\n   \"num_unique_values\": 219,\n
\"samples\": [\n   287,\n   216\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"fbs\",\n   \"properties\": {\n
\"dtype\": \"boolean\",\n   \"num_unique_values\": 2,\n
\"samples\": [\n   false,\n   true\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"restecg\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
3,\n   \"samples\": [\n   \"lv hypertrophy\",\n
\"normal\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"thalch\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 25,\n   \"min\": 60,\n   \"max\": 202,\n
\"num_unique_values\": 118,\n   \"samples\": [\n   188,\n
176\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"exang\",\n   \"properties\": {\n   \"dtype\": \"category\",\
n   \"num_unique_values\": 3,\n   \"samples\": [\n
false,\n   true\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"oldpeak\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 1.0938749708784534,\n   \"min\": -2.6,\n
\"max\": 6.2,\n   \"num_unique_values\": 53,\n
\"samples\": [\n   1.8,\n   5.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"slope\",\n   \"properties\": {\
n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n
\"samples\": [\n   \"downsloping\",\n   \"flat\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"thal\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
3,\n   \"samples\": [\n   \"fixed defect\",\n
\"normal\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"num\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1,\n   \"min\": 0,\n   \"max\": 4,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   1,\n
4\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

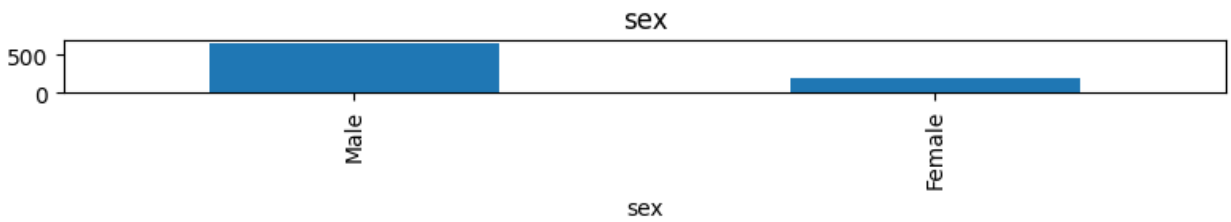
```
df["slope"].value_counts()
```

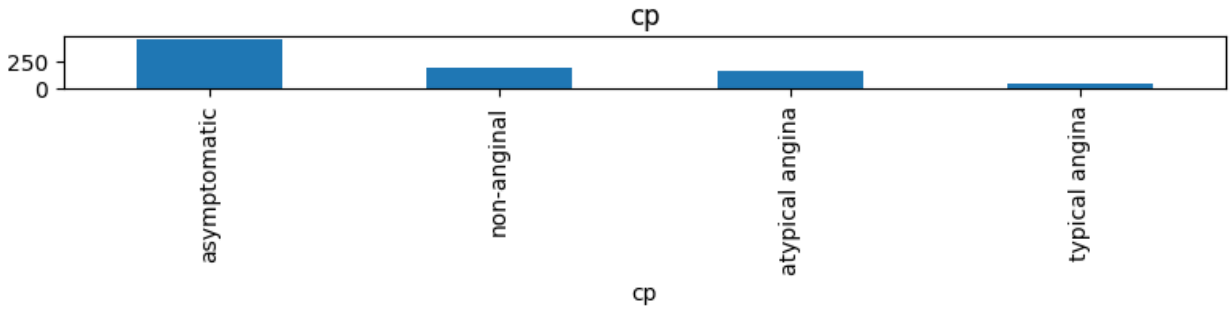
```
slope
flat           425
upsloping      276
downsloping    145
Name: count, dtype: int64
```

```
df["thal"].value_counts()
```

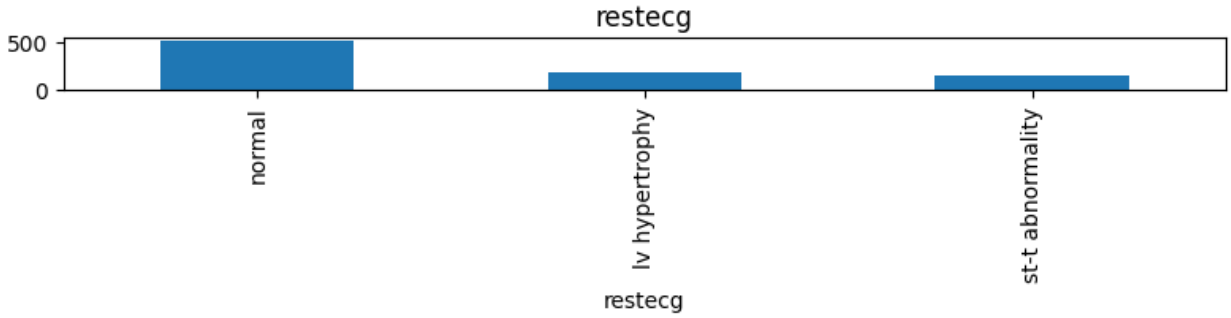
```
thal
normal               364
reversable defect    324
fixed defect         158
Name: count, dtype: int64
```

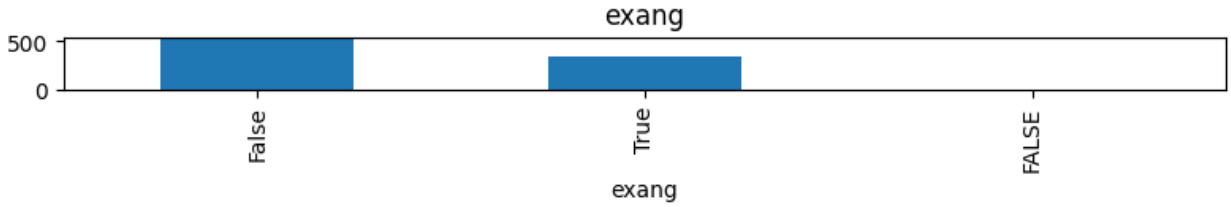
```
categorical_columns = ['sex', 'cp', 'fbs', 'restecg', 'exang',
'slope', 'thal']

fig, axs = plt.subplots(len(categorical_columns), 1, figsize=(8, 15))

# Plot bar plots for each categorical variable
for i, col in enumerate(categorical_columns):
    df[col].value_counts().plot(kind='bar', ax=axs[i])
    axs[i].set_title(col)

plt.tight_layout()
plt.show()
```

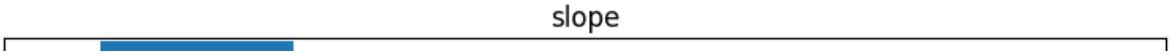

sex
500
0
Male
Female
sex


cp
250
0
asymptomatic
non-anginal
atypical angina
typical angina
cp


fbs
500
0
False
True
fbs


restecg
500
0
normal
lv hypertrophy
st-t abnormality
restecg


exang
500
0
False
True
FALSE
exang


slope

```
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 846,\n  \"fields\": [\n    {\n      \"column\": \"age\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 9,\n        \"min\": 29,\n        \"max\": 77,\n        \"num_unique_values\": 49,\n        \"samples\": [\n          76,\n          73,\n          32\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"sex\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 2,\n        \"samples\": [\n          \"Female\",\n          \"Male\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"cp\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          \"atypical angina\",\n          \"non-anginal\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"trestbps\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 19,\n        \"min\": 0,\n        \"max\": 200,\n        \"num_unique_values\": 63,\n        \"samples\": [\n          0,\n          116\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 111,\n        \"min\": 0,\n        \"max\": 603,\n        \"num_unique_values\": 219,\n        \"samples\": [\n          287,\n          216\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"fbs\",\n      \"properties\": {\n        \"dtype\": \"boolean\",\n        \"num_unique_values\": 2,\n        \"samples\": [\n          false,\n          true\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"restecg\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n        \"samples\": [\n          \"lv hypertrophy\",\n          \"normal\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"thalch\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 25,\n        \"min\": 60,\n        \"max\": 202,\n        \"num_unique_values\": 118,\n        \"samples\": [\n          188,\n          176\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"exang\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n        \"samples\": [\n          false,\n          true\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"oldpeak\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 1.0938749708784534,\n        \"min\": -2.6,\n        \"max\": 6.2,\n        \"num_unique_values\": 53,\n        \"samples\": [\n          1.8,\n          5.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": \"slope\",\n      \"properties\": {\
n        \"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"downsloping\",\n          \"flat\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"thal\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
3,\n        \"samples\": [\n          \"fixed defect\",\n
\"normal\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"num\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 0,\n        \"max\": 4,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          1,\n
4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


```
numerical_features = ['age', 'trestbps','chol','thalch','oldpeak']

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[numerical_features] = scaler.fit_transform(df[numerical_features])

df.shape
```

```
(846, 13)
```

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Convert 'exang' column to string type before applying Label Encoding
df['exang'] = df['exang'].astype(str)

# Apply Label Encoding
LE = LabelEncoder()
df["sex"] = LE.fit_transform(df["sex"])
df["cp"] = LE.fit_transform(df["cp"])
df["fbs"] = LE.fit_transform(df["fbs"])
df["restecg"] = LE.fit_transform(df["restecg"])
df["exang"] = LE.fit_transform(df["exang"]) # Now this line should
work without error
df["slope"] = LE.fit_transform(df["slope"])
df["thal"] = LE.fit_transform(df["thal"])

df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 846,\n  \"fields\": [\
n    {\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.0005915410159436,\n
\"min\": -2.6859355916955674,\n        \"max\": 2.5923489857131434,\n
\"num_unique_values\": 49,\n        \"samples\": [\n
2.4823847236837953,\n          2.1524919375957507,\n          -
```

```
2.356042805607523\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sex\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 3,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          1,\n
2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"trestbps\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.0005915410159456,\n        \"min\": -
6.886812293479787,\n        \"max\": 3.512365231747607,\n
\"num_unique_values\": 63,\n        \"samples\": [\n          -
6.886812293479787,\n          -0.8552893288478985\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.0005915410159465,\n
\"min\": -1.8160767112378515,\n        \"max\": 3.5877271041039225,\n
\"num_unique_values\": 219,\n        \"samples\": [\n
0.7558829819679348,\n          0.11961420769054522\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fbs\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"restecg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 2,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"thalch\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.0005915410159452,\n        \"min\": -
3.015158813569593,\n        \"max\": 2.5331693922961986,\n
\"num_unique_values\": 118,\n        \"samples\": [\n
1.9861511184784444,\n          1.5172783123489408\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"exang\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\":
0,\n        \"max\": 2,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          1,\n          2\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"oldpeak\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
1.0005915410159434,\n        \"min\": -3.193526012938529,\n
\"max\": 4.856027908081501,\n        \"num_unique_values\": 53,\n
```

\"samples\": [\n   0.8312509475714859,\n
3.758361464306042\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"slope\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0,\n   \"min\": 0,\n   \"max\": 2,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   0,\n
1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"thal\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0,\n   \"min\": 0,\n   \"max\": 2,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   0,\n
1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"num\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1,\n   \"min\": 0,\n   \"max\": 4,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   1,\n
4\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"df"}

```
x =
df[["age","sex","cp","trestbps","chol","fbs","restecg","thalch","exang
","oldpeak","slope","thal"]]
y = df["num"]
```

```
x.head()
```

{"summary":"{\n \"name\": \"x\",\n \"rows\": 846,\n \"fields\": [\n
{\n   \"column\": \"age\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 1.0005915410159436,\n
\"min\": -2.6859355916955674,\n   \"max\": 2.5923489857131434,\n
\"num_unique_values\": 49,\n   \"samples\": [\n
2.4823847236837953,\n   2.1524919375957507,\n   -
2.356042805607523\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"sex\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   0,\n
1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"cp\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0,\n   \"min\": 0,\n   \"max\": 3,\n
\"num_unique_values\": 4,\n   \"samples\": [\n   1,\n
2\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"trestbps\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 1.0005915410159456,\n   \"min\": -
6.886812293479787,\n   \"max\": 3.512365231747607,\n
\"num_unique_values\": 63,\n   \"samples\": [\n   -
6.886812293479787,\n   -0.8552893288478985\n   ],\n

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.0005915410159465,\n
\"min\": -1.8160767112378515,\n        \"max\": 3.5877271041039225,\n
\"num_unique_values\": 219,\n        \"samples\": [\n
0.7558829819679348,\n          0.11961420769054522\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fbs\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"restecg\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 2,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"thalch\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.0005915410159452,\n        \"min\": -
3.015158813569593,\n        \"max\": 2.5331693922961986,\n
\"num_unique_values\": 118,\n        \"samples\": [\n
1.9861511184784444,\n          1.5172783123489408\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"exang\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 0,\n        \"min\":
0,\n        \"max\": 2,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          1,\n          2\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"oldpeak\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
1.0005915410159434,\n        \"min\": -3.193526012938529,\n
\"max\": 4.856027908081501,\n        \"num_unique_values\": 53,\n
\"samples\": [\n          0.8312509475714859,\n
3.758361464306042\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"slope\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 2,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"thal\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 2,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"x"}
```


```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.3, random_state=23)
```

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

```
y_pred_train = dt.predict(x_train)
y_pred_test = dt.predict(x_test)
```

```
from sklearn.metrics import
accuracy_score,precision_score,f1_score,recall_score
```

```
from sklearn.metrics import
accuracy_score,precision_score,f1_score,recall_score
```

```
ac1 = accuracy_score(y_train,y_pred_train)
print("Training accuracy:", round(ac1, 2)) # Use the round() function
instead of the method
```

```
ac2 = accuracy_score(y_test,y_pred_test)
print("Test accuracy:", round(ac2, 2)) # Use the round() function
instead of the methodz
```

```
Training accuracy: 1.0
Test accuracy: 0.52
```

```
# prompt: tell me some other hyper paramters which can be improves the
model performance for the model above dt
```

```
# - `max_depth`: This parameter controls the maximum depth of the
decision tree. A larger value allows the tree to grow deeper,
potentially capturing more complex relationships in the data, but also
increasing the risk of overfitting.
# - `min_samples_split`: This parameter controls the minimum number of
samples required to split a node. A larger value makes the tree more
conservative, reducing the risk of overfitting but potentially missing
out on some important relationships in the data.
# - `min_samples_leaf`: This parameter controls the minimum number of
samples required in a leaf node. A larger value makes the tree more
conservative, reducing the risk of overfitting but potentially leading
to underfitting.
# - `max_features`: This parameter controls the maximum number of
features considered when splitting a node. A smaller value reduces the
risk of overfitting by making the tree more focused on the most
important features.
# - `criterion`: This parameter controls the function used to evaluate
the quality of a split. The most common options are "gini" and
"entropy".
# - `splitter`: This parameter controls the strategy used to split a
node. The most common options are "best" and "random".
```

```
# Change the min_samples_split parameter to 5
dt =
DecisionTreeClassifier(min_samples_leaf=25,max_depth=10,max_features=1
3,criterion='gini',splitter='random')

# Fit the model
dt.fit(x_train, y_train)

# Predict on the training and test sets
y_pred_train = dt.predict(x_train)
y_pred_test = dt.predict(x_test)

# Evaluate the model's performance
ac1 = accuracy_score(y_train, y_pred_train)
ac2 = accuracy_score(y_test, y_pred_test)

# Print the results
# Use the round() function to round the accuracy scores
print("Training accuracy:", round(ac1, 2))
print("Test accuracy:", round(ac2, 2))
```

```
Training accuracy: 0.6
Test accuracy: 0.51
```

```
# Create a new decision tree classifier with max_depth set to 5
dt = DecisionTreeClassifier(max_depth=6)

# Fit the new decision tree classifier on the training data
dt.fit(x_train, y_train)

# Predict the class labels for the training and test data
y_pred_train = dt.predict(x_train)
y_pred_test = dt.predict(x_test)

# Calculate the accuracy score for the training and test data
ac1 = accuracy_score(y_train, y_pred_train)
print("Training accuracy:", round(ac1, 2)) # Use the round() function
to round the float

ac2 = accuracy_score(y_test, y_pred_test)
print("Test accuracy:", round(ac2, 2)) # Use the round() function to
round the float
```

```
Training accuracy: 0.73
Test accuracy: 0.52
```

```
# Calculate the accuracy score for the training and test data
ac1 = accuracy_score(y_train, y_pred_train)
print("Training accuracy:", round(ac1, 2))
```

```
ac2 = accuracy_score(y_test, y_pred_test)
print("Test accuracy:", round(ac2, 2))
```

```
Training accuracy: 0.59
Test accuracy: 0.56
```

```
!pip install graphviz
from sklearn.tree import export_graphviz
import graphviz

dot_data = export_graphviz(dt_best, out_file=None,
feature_names=x.columns)
graph = graphviz.Source(dot_data)
graph
```

```
Requirement already satisfied: graphviz in
/usr/local/lib/python3.10/dist-packages (0.20.3)
```

```
# Calculate precision
precision = precision_score(y_train, y_pred_train, average='macro')
print("Precision:", precision)

precision = precision_score(y_test, y_pred_test, average='macro')
print("Precision:", precision)
```

```
Precision: 0.23199412951751971
Precision: 0.21582878302893366
```

```
# Calculate recall
recall = recall_score(y_train, y_pred_train, average='macro')
print("Recall:", recall)

recall = recall_score(y_test, y_pred_test, average='macro')
print("Recall:", recall)
```

```
Recall: 0.3119790997473125
Recall: 0.2871739130434783
```

```
# Calculate F1-score
f1 = f1_score(y_train, y_pred_train, average='macro')
print("F1-score:", f1)

f1 = f1_score(y_test, y_pred_test, average='macro')
print("F1-score:", f1)
```

```
F1-score: 0.26534511547053574
F1-score: 0.24489916214054147
```

```
from sklearn.metrics import roc_curve,roc_auc_score

df["Y_proba"] = dt_best.predict_proba(x)[:,1]
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 846,\n  \"fields\": [\
n    {\n      \"column\": \"age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.0005915410159436,\n
\"min\": -2.6859355916955674,\n        \"max\": 2.5923489857131434,\n
\"num_unique_values\": 49,\n        \"samples\": [\n
2.4823847236837953,\n          2.1524919375957507,\n          -
2.356042805607523\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sex\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 3,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          1,\n
2\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"trestbps\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.0005915410159456,\n        \"min\": -
6.886812293479787,\n        \"max\": 3.512365231747607,\n
\"num_unique_values\": 63,\n        \"samples\": [\n          -
6.886812293479787,\n          -0.8552893288478985\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"chol\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.0005915410159465,\n
\"min\": -1.8160767112378515,\n        \"max\": 3.5877271041039225,\n
\"num_unique_values\": 219,\n        \"samples\": [\n
0.7558829819679348,\n          0.11961420769054522\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"fbs\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          0,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
```

\"column\": \"restecg\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 2,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"thalch\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1.0005915410159452,\n   \"min\": -3.015158813569593,\n   \"max\": 2.5331693922961986,\n   \"num_unique_values\": 118,\n   \"samples\": [\n   1.9861511184784444,\n   1.5172783123489408\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"exang\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 2,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   1,\n   2\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"oldpeak\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1.0005915410159434,\n   \"min\": -3.193526012938529,\n   \"max\": 4.856027908081501,\n   \"num_unique_values\": 53,\n   \"samples\": [\n   0.8312509475714859,\n   3.758361464306042\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"slope\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 2,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"thal\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 2,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"num\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 0,\n   \"max\": 4,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   1,\n   4\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Y_proba\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.17811714754879845,\n   \"min\": 0.043478260869565216,\n   \"max\": 0.6029411764705882,\n   \"num_unique_values\": 13,\n   \"samples\": [\n   0.4883720930232558,\n   0.125\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
# prompt: print the number of nodes and depth of tree

print("The number of nodes in the tree is:", dt_best.tree_.node_count)
print("The depth of the tree is:", dt_best.tree_.max_depth)
```

```
The number of nodes in the tree is: 25
The depth of the tree is: 5
```

```
# bagging classifer
# cross validation method
from sklearn.ensemble import BaggingClassifier
dt_model = BaggingClassifier(max_features=0.7,max_samples=0.6)

from sklearn.model_selection import train_test_split

training_accuracy = []
test_accuracy = []
for i in range(1,101):
    X_train,X_test,Y_train,Y_test = train_test_split(x,y,
test_size=0.30,random_state=i)
    dt_model.fit(X_train,Y_train)
    Y_pred_train = dt_model.predict(X_train)
    Y_pred_test = dt_model.predict(X_test)
    training_accuracy.append(accuracy_score(Y_train,Y_pred_train))
    test_accuracy.append(accuracy_score(Y_test,Y_pred_test))

import numpy as np
print("Cross validation Training score: ",
np.mean(training_accuracy).round(2))
print("Cross validation Test score: ",
np.mean(test_accuracy).round(2))
```

```
Cross validation Training score:  0.92
Cross validation Test score:  0.55
```

```
# random forest classifer
# cross validation method
from sklearn.ensemble import RandomForestClassifier
dt_model = RandomForestClassifier(max_features=0.7,max_samples=0.6)

from sklearn.model_selection import train_test_split

training_accuracy = []
test_accuracy = []
for i in range(1,101):
    X_train,X_test,Y_train,Y_test = train_test_split(x,y,
test_size=0.30,random_state=i)
    dt_model.fit(X_train,Y_train)
    Y_pred_train = dt_model.predict(X_train)
    Y_pred_test = dt_model.predict(X_test)
    training_accuracy.append(accuracy_score(Y_train,Y_pred_train))
    test_accuracy.append(accuracy_score(Y_test,Y_pred_test))

import numpy as np
print("Cross validation Training score: ",
np.mean(training_accuracy).round(2))
```

```
print("Cross validation Test score: ",
np.mean(test_accuracy).round(2))
```

```
Cross validation Training score:  0.98
Cross validation Test score:  0.57
```

```
# adaboost classifer
# cross validation method
from sklearn.ensemble import AdaBoostClassifier
dt_model = AdaBoostClassifier(n_estimators=500,learning_rate=0.1)

from sklearn.model_selection import train_test_split

training_accuracy = []
test_accuracy = []
for i in range(1,101):
    X_train,X_test,Y_train,Y_test = train_test_split(x,y,
test_size=0.30,random_state=i)
    dt_model.fit(X_train,Y_train)
    Y_pred_train = dt_model.predict(X_train)
    Y_pred_test = dt_model.predict(X_test)
    training_accuracy.append(accuracy_score(Y_train,Y_pred_train))
    test_accuracy.append(accuracy_score(Y_test,Y_pred_test))

import numpy as np
print("Cross validation Training score: ",
np.mean(training_accuracy).round(2))
print("Cross validation Test score: ",
np.mean(test_accuracy).round(2))

# prompt: fit the above data using xgb classifier and give the best
test accuracy by trying with different parameters

# xgboost classifier
# cross validation method
from xgboost import XGBClassifier
dt_model = XGBClassifier(max_depth=5, learning_rate=0.1,
n_estimators=500,reg_lambda=1,gamma=10)


training_accuracy = []
test_accuracy = []
for i in range(1,101):
    X_train,X_test,Y_train,Y_test = train_test_split(x,y,
test_size=0.30,random_state=i)
    dt_model.fit(X_train,Y_train)
    Y_pred_train = dt_model.predict(X_train)
    Y_pred_test = dt_model.predict(X_test)
    training_accuracy.append(accuracy_score(Y_train,Y_pred_train))
    test_accuracy.append(accuracy_score(Y_test,Y_pred_test))
```

```
print("Cross validation Training score: ",
np.mean(training_accuracy).round(2))
print("Cross validation Test score: ",
np.mean(test_accuracy).round(2))

# prompt: with out using cross validation tune the parameters for XGB
classifier such as reg_lambda, learning_rate, gamma

dt_model = XGBClassifier(max_depth=5, learning_rate=0.1,
n_estimators=500,reg_lambda=1,gamma=10)
dt_model.fit(x_train,y_train)
y_pred_train = dt_model.predict(x_train)
y_pred_test = dt_model.predict(x_test)

ac1 = accuracy_score(y_train,y_pred_train)
print("Training accuracy:", ac1.round(2))

ac2 = accuracy_score(y_test,y_pred_test)
print("Test accuracy:", ac2.round(2))

'''
# Calculate precision
precision = precision_score(y_train, y_pred_train, average='macro')
print("Precision:", precision)

precision = precision_score(y_test, y_pred_test, average='macro')
print("Precision:", precision)
# Calculate recall
recall = recall_score(y_train, y_pred_train, average='macro')
print("Recall:", recall)

recall = recall_score(y_test, y_pred_test, average='macro')
print("Recall:", recall)
# Calculate F1-score
f1 = f1_score(y_train, y_pred_train, average='macro')
print("F1-score:", f1)

f1 = f1_score(y_test, y_pred_test, average='macro')
print("F1-score:", f1)
'''

# prompt: change the parameters using grid search cv and fit the model
once again give the best accracy

from sklearn.model_selection import GridSearchCV
param_grid = {
    'max_depth': [3, 5, 7, 10],
    'learning_rate': [0.01, 0.1, 0.2, 0.3],
    'n_estimators': [100, 150, 200, 500],
    'reg_lambda': [0.1, 1, 10],
```

```
    'gamma': [1, 10]
}

grid_search = GridSearchCV(XGBClassifier(), param_grid, cv=5)
grid_search.fit(x_train, y_train)

best_params = grid_search.best_params_
print("Best parameters:", best_params)

best_model = XGBClassifier(**best_params)
best_model.fit(x_train, y_train)

y_pred_test = best_model.predict(x_test)

accuracy = accuracy_score(y_test, y_pred_test)
print("Test accuracy with best parameters:", accuracy.round(2))
```