IMPORT NECESSARY LIBRARIES

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import warnings
warnings.filterwarnings('ignore')
```

IMPORT MODULES

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import
accuracy_score,classification_report,confusion_matrix,r2_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

LOAD THE DATASET

```
data=pd.read_csv(r"C:\Users\Admin\Downloads\processed_cleveland.csv")
data.head()
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
slope  \
0   63    1   1       145   233    1        2      150      0      2.3
3
1   67    1   4       160   286    0        2      108      1      1.5
2
2   67    1   4       120   229    0        2      129      1      2.6
2
3   37    1   3       130   250    0        0      187      0      3.5
3
4   41    0   2       130   204    0        2      172      0      1.4
1

  ca thal  num
0  0    6    0
1  3    3    2
2  2    7    1
```

```
3  0    3    0
4  0    3    0
```

```
data.shape
```

```
(303, 14)
```

```
data.duplicated().sum()
```

```
0
```

```
data.isnull().sum()
```

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

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    object 
 12  thal      303 non-null    object 
 13  num       303 non-null    int64  
```

```
dtypes: float64(1), int64(11), object(2)
memory usage: 33.3+ KB
```

```
data['ca']=pd.to_numeric(data['ca'],errors='coerce')
data['thal']=pd.to_numeric(data['thal'],errors='coerce')

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   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    float64
 13  num       303 non-null    int64
dtypes: float64(3), int64(11)
memory usage: 33.3 KB
```

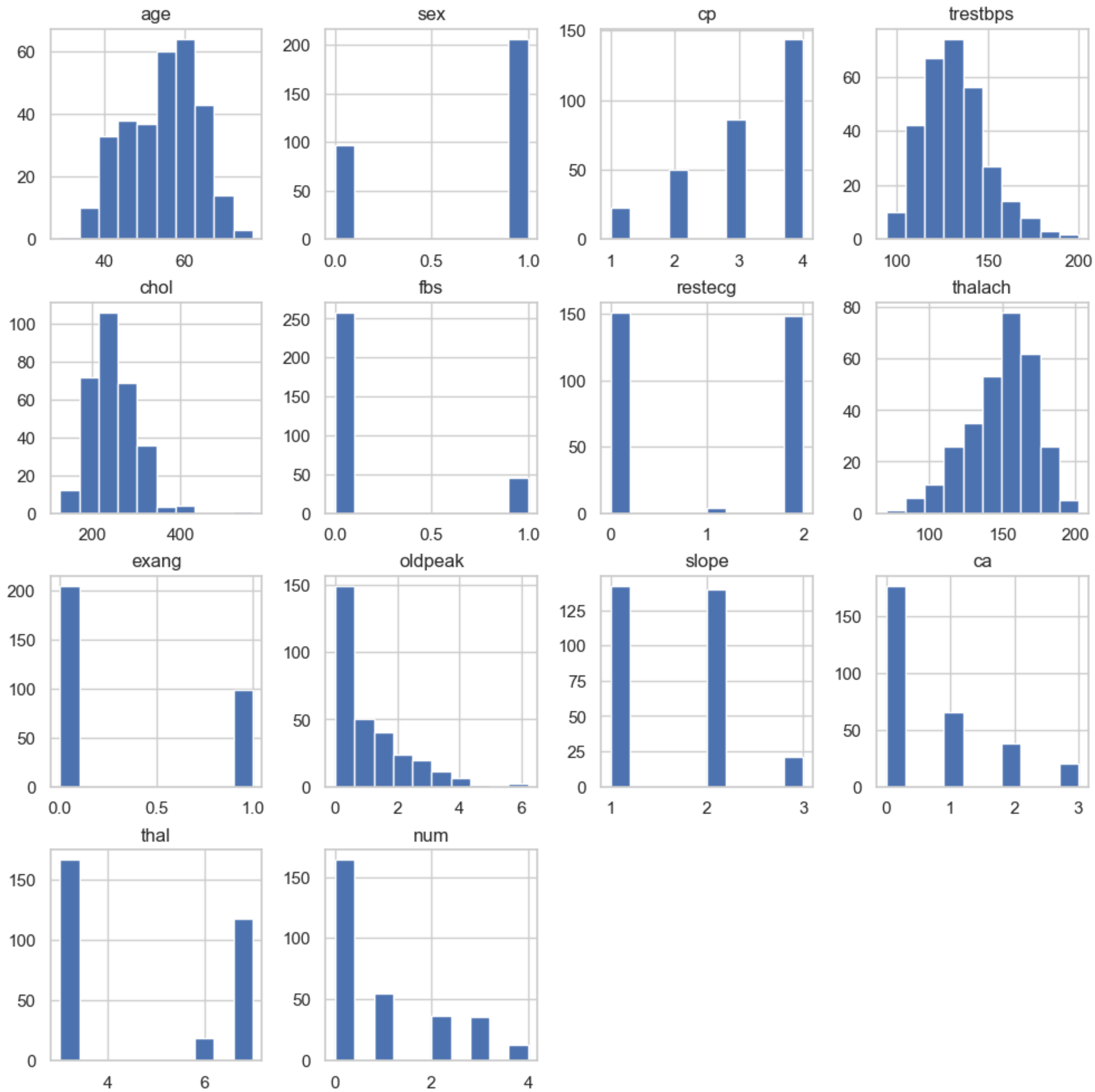
```
data.describe()
```

```
              age         sex          cp    trestbps        chol
fbs  \
count  303.000000  303.000000  303.000000  303.000000  303.000000
303.000000
mean    54.438944    0.679868    3.158416  131.689769  246.693069
0.148515
std      9.038662    0.467299    0.960126   17.599748   51.776918
0.356198
min     29.000000    0.000000    1.000000   94.000000  126.000000
0.000000
25%     48.000000    0.000000    3.000000  120.000000  211.000000
0.000000
50%     56.000000    1.000000    3.000000  130.000000  241.000000
0.000000
75%     61.000000    1.000000    4.000000  140.000000  275.000000
0.000000
max     77.000000    1.000000    4.000000  200.000000  564.000000
1.000000
```

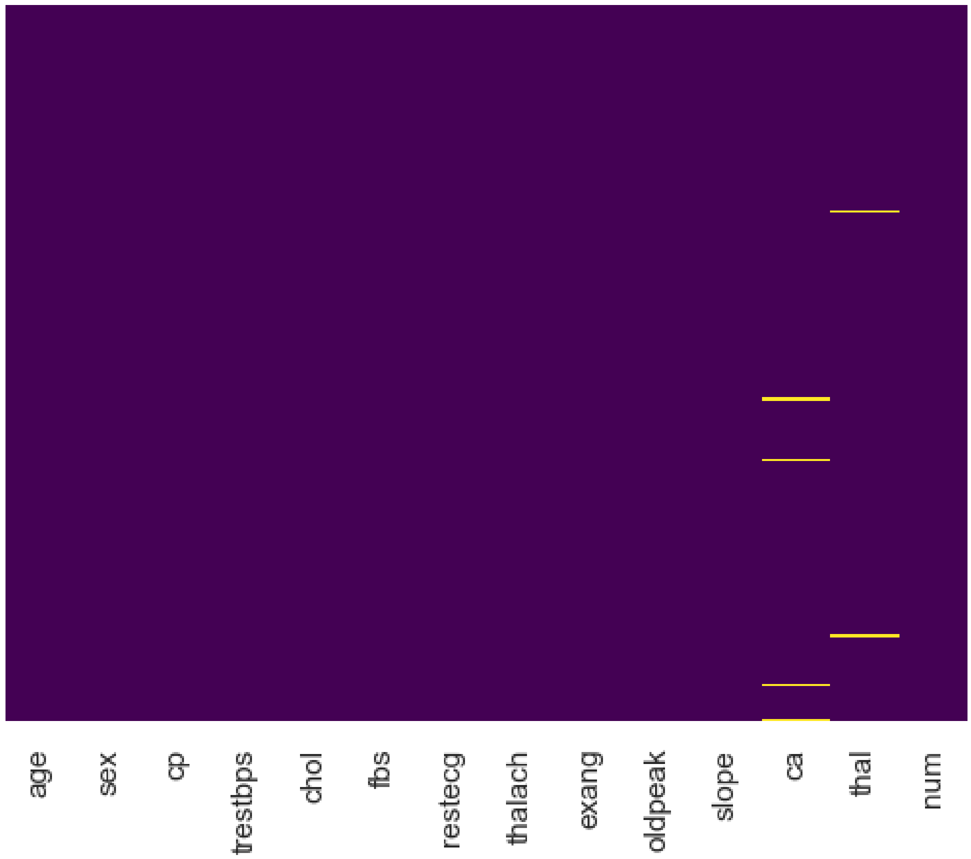
```
         restecg    thalach       exang     oldpeak       slope
ca  \
count  303.000000  303.000000  303.000000  303.000000  303.000000
299.000000
mean     0.990099  149.607261    0.326733    1.039604    1.600660
0.672241
std      0.994971   22.875003    0.469794    1.161075    0.616226
0.937438
min      0.000000   71.000000    0.000000    0.000000    1.000000
0.000000
25%      0.000000  133.500000    0.000000    0.000000    1.000000
0.000000
50%      1.000000  153.000000    0.000000    0.800000    2.000000
0.000000
75%      2.000000  166.000000    1.000000    1.600000    2.000000
1.000000
max      2.000000  202.000000    1.000000    6.200000    3.000000
3.000000

             thal         num
count  301.000000  303.000000
mean     4.734219    0.937294
std      1.939706    1.228536
min      3.000000    0.000000
25%      3.000000    0.000000
50%      3.000000    0.000000
75%      7.000000    2.000000
max      7.000000    4.000000
```

VISUALIZING THE DATA

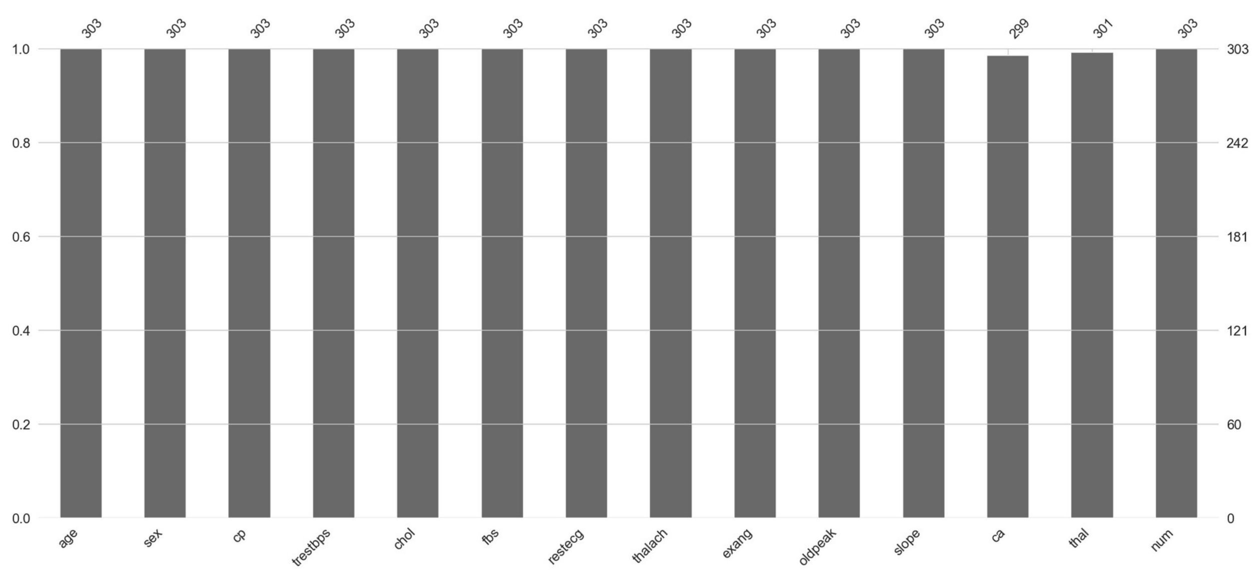
```
data.hist(figsize=(12,12))
plt.show()
```



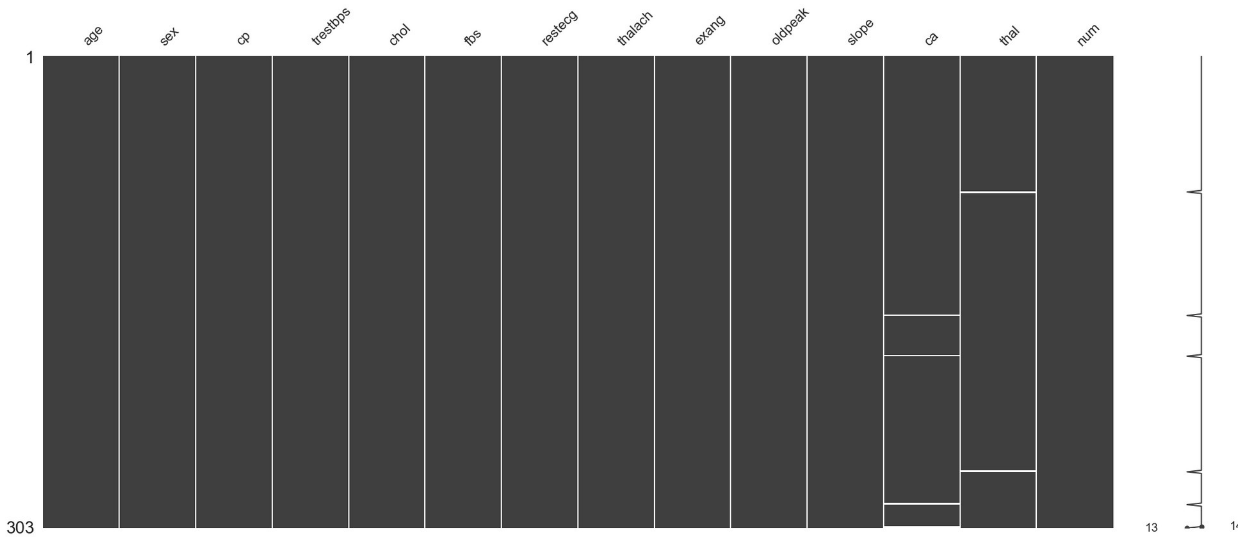

```
sns.heatmap(data.isnull(),yticklabels=False,cbar=False,cmap='viridis')
plt.show()
```




```
msno.bar(data)

<AxesSubplot:>
```



```
msno.matrix(data)
plt.show()
```




```
data.corr()
```

```
               age       sex        cp  trestbps      chol
fbs  \
age       1.000000 -0.097542  0.104139  0.284946  0.208950  0.118530

sex      -0.097542  1.000000  0.010084 -0.064456 -0.199915  0.047862

cp        0.104139  0.010084  1.000000 -0.036077  0.072319 -0.039975

trestbps  0.284946 -0.064456 -0.036077  1.000000  0.130120  0.175340

chol      0.208950 -0.199915  0.072319  0.130120  1.000000  0.009841

fbs       0.118530  0.047862 -0.039975  0.175340  0.009841  1.000000

restecg   0.148868  0.021647  0.067505  0.146560  0.171043  0.069564

thalach  -0.393806 -0.048663 -0.334422 -0.045351 -0.003432 -0.007854

exang     0.091661  0.146201  0.384060  0.064762  0.061310  0.025665

oldpeak   0.203805  0.102173  0.202277  0.189171  0.046564  0.005747

slope     0.161770  0.037533  0.152050  0.117382 -0.004062  0.059894

ca        0.362605  0.093185  0.233214  0.098773  0.119000  0.145478

thal      0.127389  0.380936  0.265246  0.133554  0.014214  0.071358
```

```
num        0.222853  0.224469  0.407075  0.157754  0.070909  0.059186


            restecg   thalach     exang   oldpeak     slope
ca  \
age        0.148868 -0.393806  0.091661  0.203805  0.161770  0.362605

sex        0.021647 -0.048663  0.146201  0.102173  0.037533  0.093185

cp         0.067505 -0.334422  0.384060  0.202277  0.152050  0.233214

trestbps   0.146560 -0.045351  0.064762  0.189171  0.117382  0.098773

chol       0.171043 -0.003432  0.061310  0.046564 -0.004062  0.119000

fbs        0.069564 -0.007854  0.025665  0.005747  0.059894  0.145478

restecg    1.000000 -0.083389  0.084867  0.114133  0.133946  0.128343

thalach   -0.083389  1.000000 -0.378103 -0.343085 -0.385601 -0.264246

exang      0.084867 -0.378103  1.000000  0.288223  0.257748  0.145570

oldpeak    0.114133 -0.343085  0.288223  1.000000  0.577537  0.295832

slope      0.133946 -0.385601  0.257748  0.577537  1.000000  0.110119

ca         0.128343 -0.264246  0.145570  0.295832  0.110119  1.000000

thal       0.024531 -0.279631  0.329680  0.341004  0.287232  0.256382

num        0.183696 -0.415040  0.397057  0.504092  0.377957  0.518909


               thal       num
age        0.127389  0.222853
sex        0.380936  0.224469
cp         0.265246  0.407075
trestbps   0.133554  0.157754
chol       0.014214  0.070909
fbs        0.071358  0.059186
restecg    0.024531  0.183696
thalach   -0.279631 -0.415040
exang      0.329680  0.397057
oldpeak    0.341004  0.504092
slope      0.287232  0.377957
ca         0.256382  0.518909
thal       1.000000  0.509923
num        0.509923  1.000000
```

```python
data.corr()['chol']
```

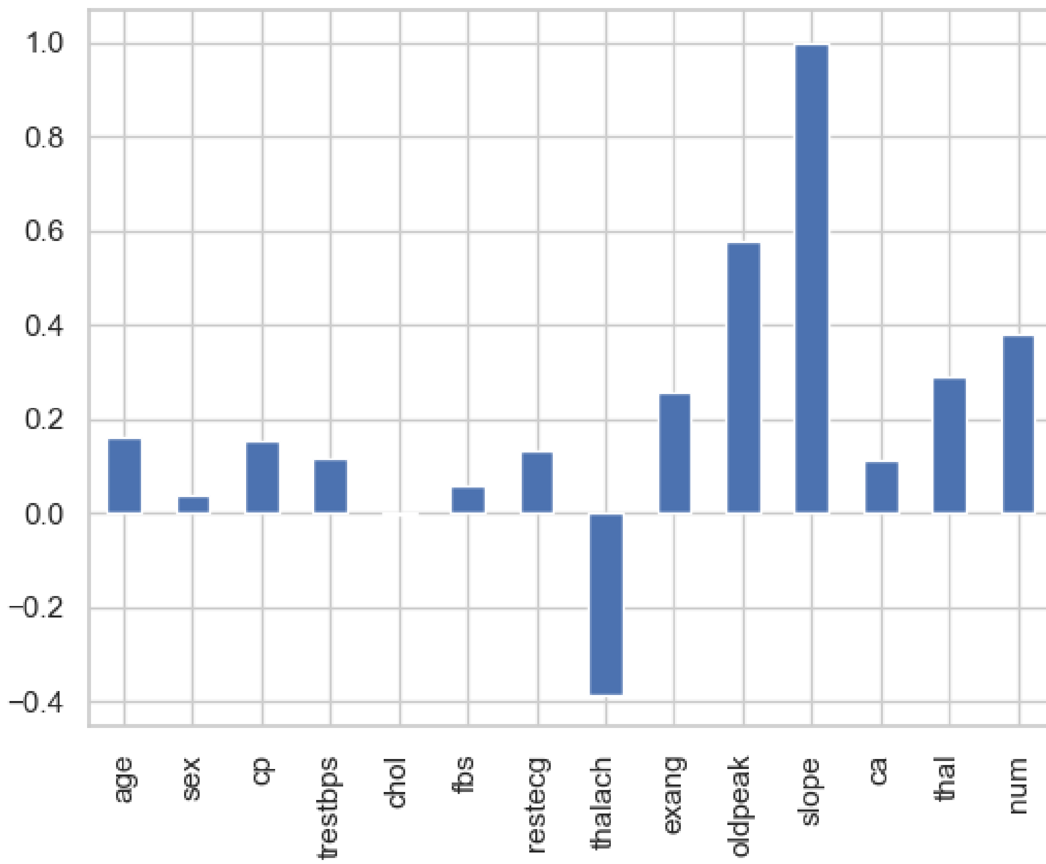
```
age         0.208950
sex        -0.199915
cp          0.072319
trestbps    0.130120
chol        1.000000
fbs         0.009841
restecg     0.171043
thalach    -0.003432
exang       0.061310
oldpeak     0.046564
slope      -0.004062
ca          0.119000
thal        0.014214
num         0.070909
Name: chol, dtype: float64
```

```
data.corr()['slope']
```

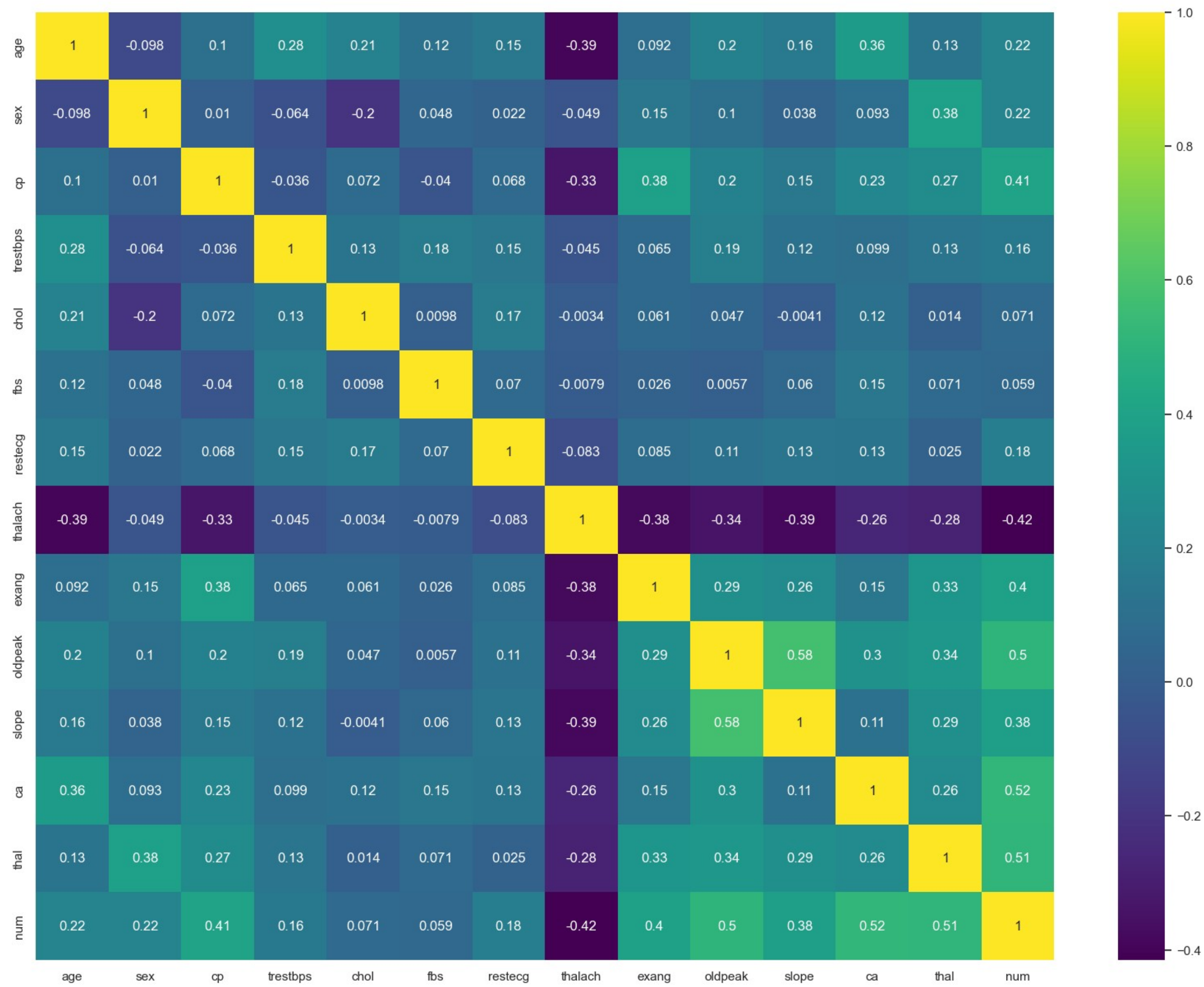
```
age         0.161770
sex         0.037533
cp          0.152050
trestbps    0.117382
chol       -0.004062
fbs         0.059894
restecg     0.133946
thalach    -0.385601
exang       0.257748
oldpeak     0.577537
slope       1.000000
ca          0.110119
thal        0.287232
num         0.377957
Name: slope, dtype: float64
```

```
data.corr()['slope'].plot(kind='bar')
```

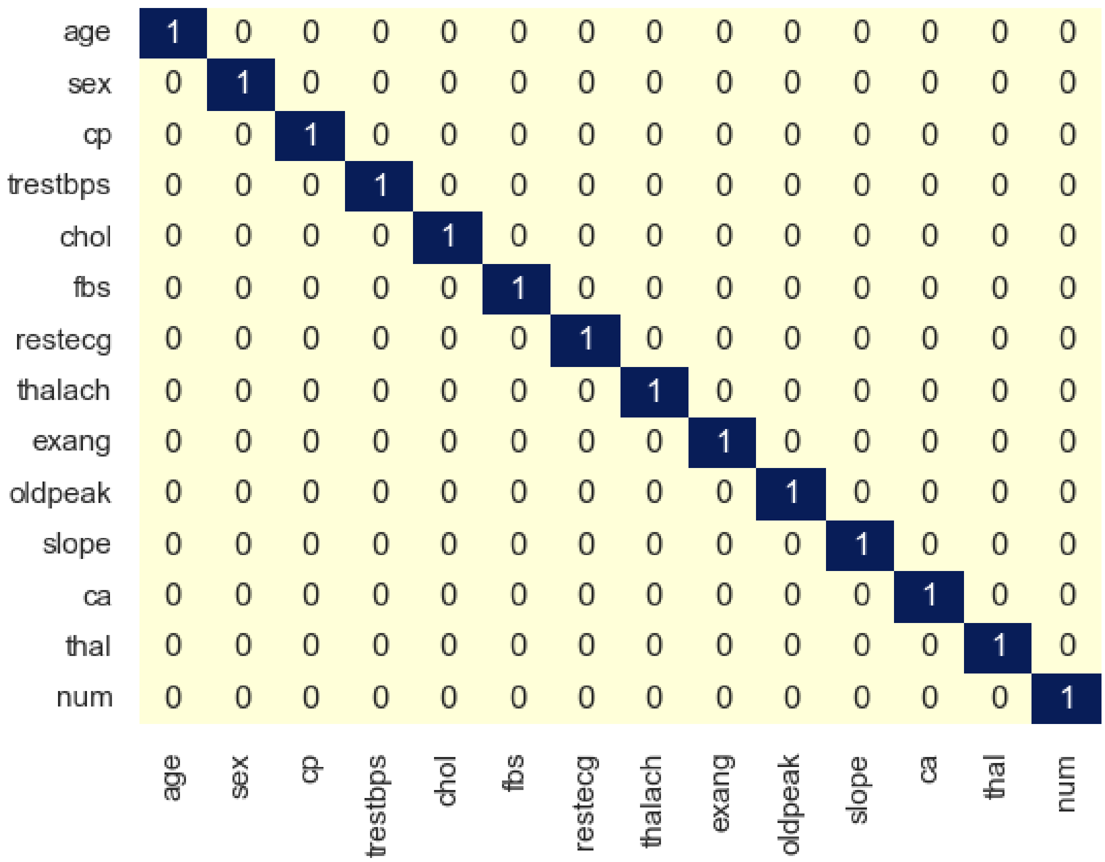
```
<AxesSubplot:>
```



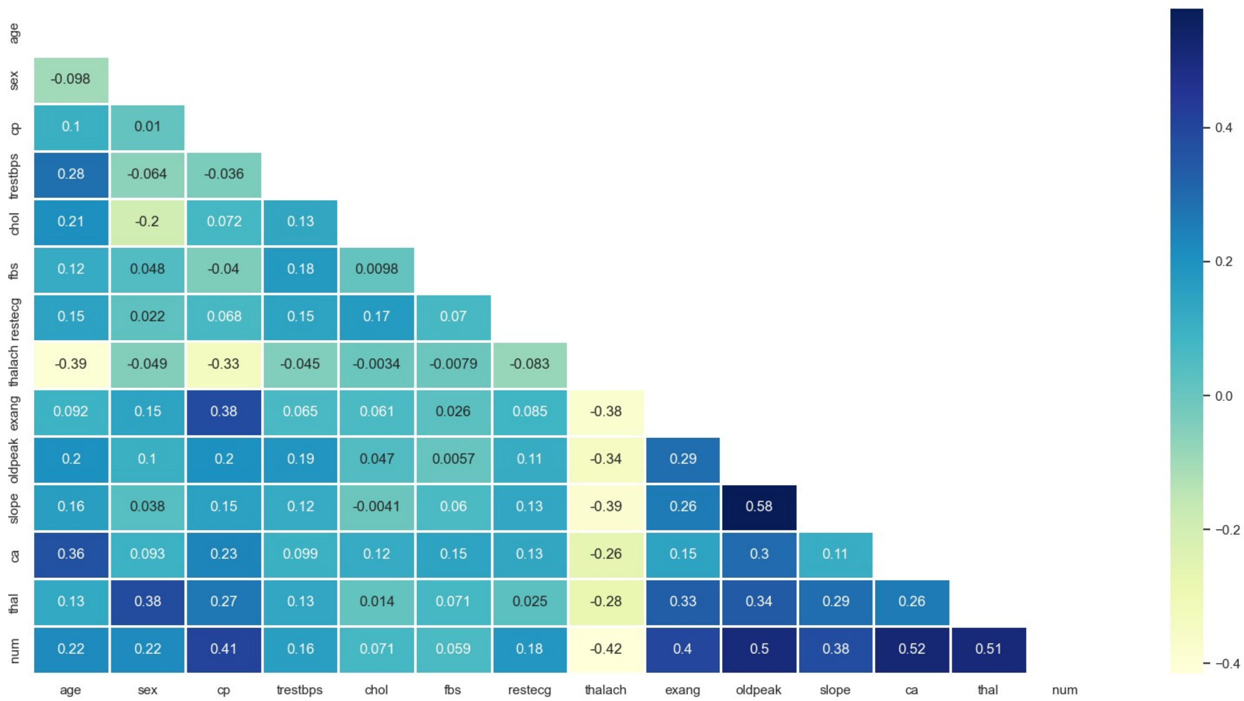

```
plt.figure(figsize=(20,15))
corr = data.corr()
sns.heatmap(data.corr(), cmap="viridis", annot=True)
plt.show()
```



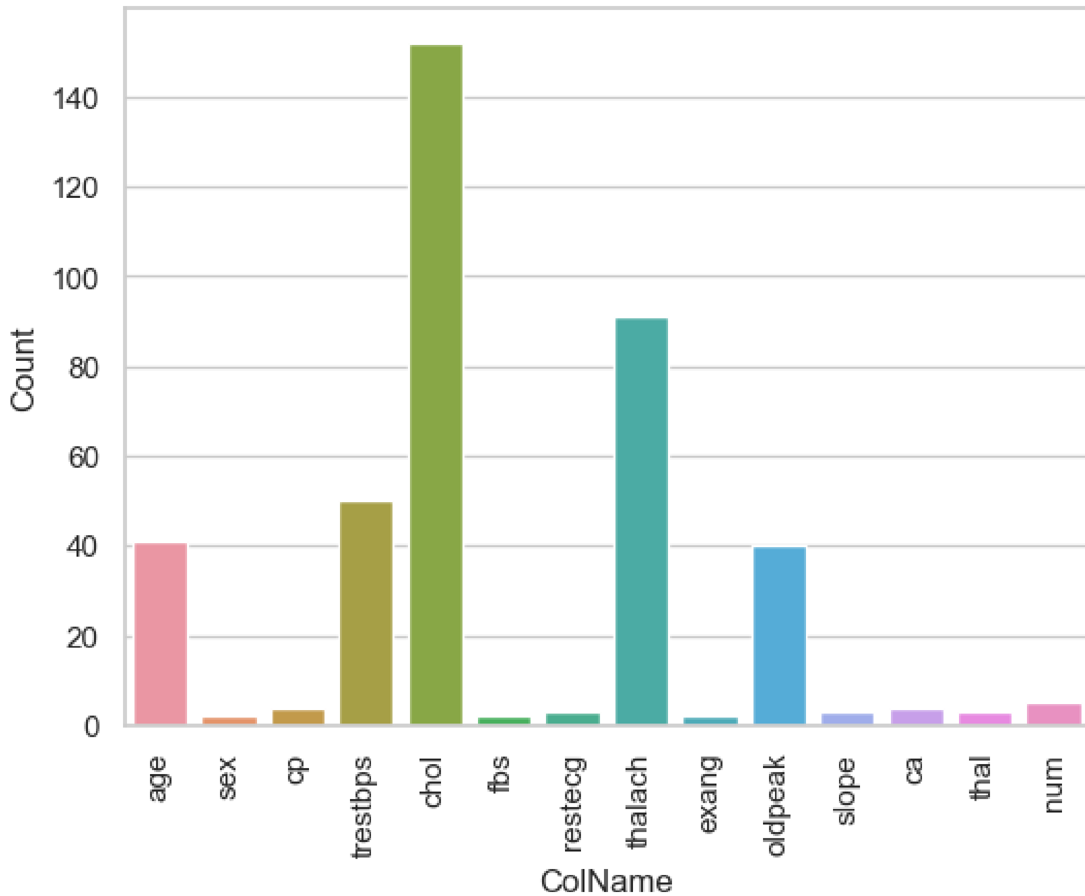

```python
sns.heatmap(data.corr() > 0.9, annot=True, cbar=False,cmap="YlGnBu")
plt.show()
```



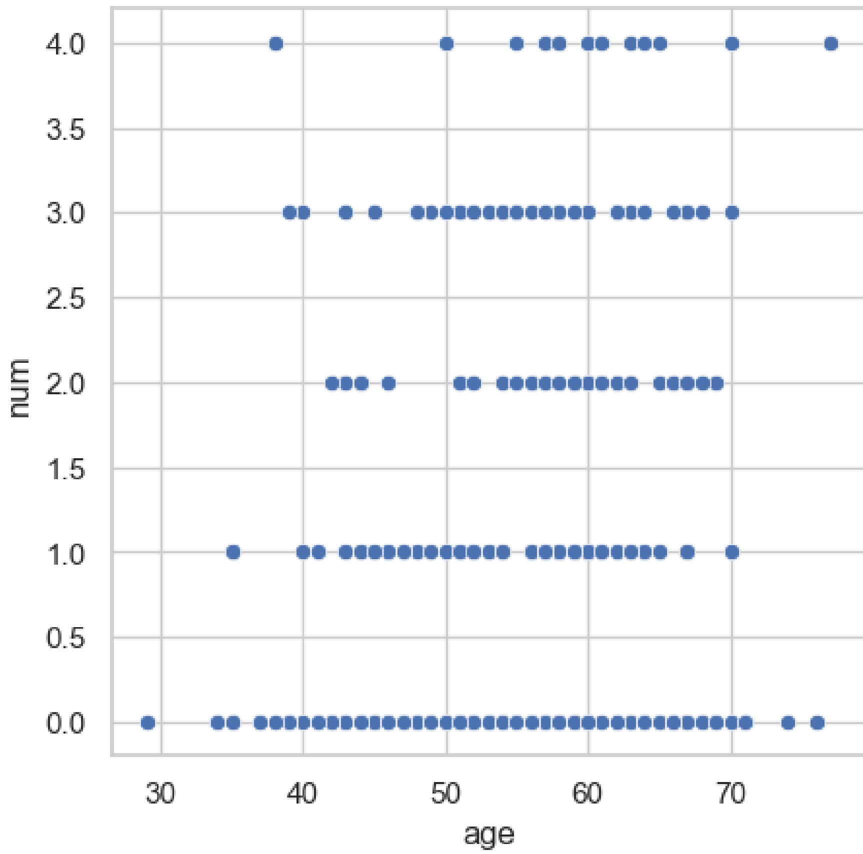

```
plt.figure(figsize=(20,10))
corr = data.corr()
mask=np.triu(np.ones_like(corr,dtype=bool))

sns.heatmap(data=corr, mask=mask,
cmap="YlGnBu",annot=True,linewidth=2)
plt.show()
```




```python
unique=data.nunique().to_frame()
unique.columns=['Count']
unique.index.names=['ColName']
unique=unique.reset_index()
sns.set(style='whitegrid',color_codes='True')
sns.barplot(x='ColName', y = 'Count', data = unique)
plt.xticks(rotation=90)
plt.show()
```

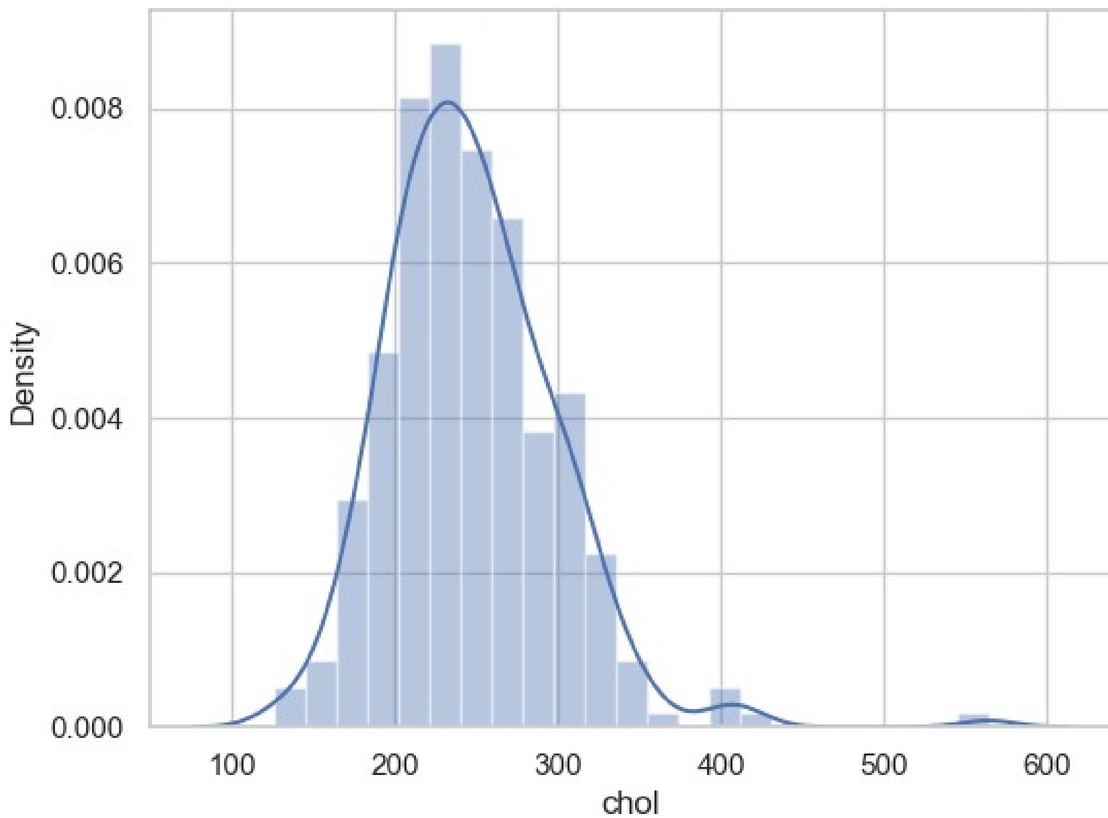



```
plt.figure(figsize=(5,5))
sns.scatterplot(x=data['age'],y=data['num'])
plt.show()
```

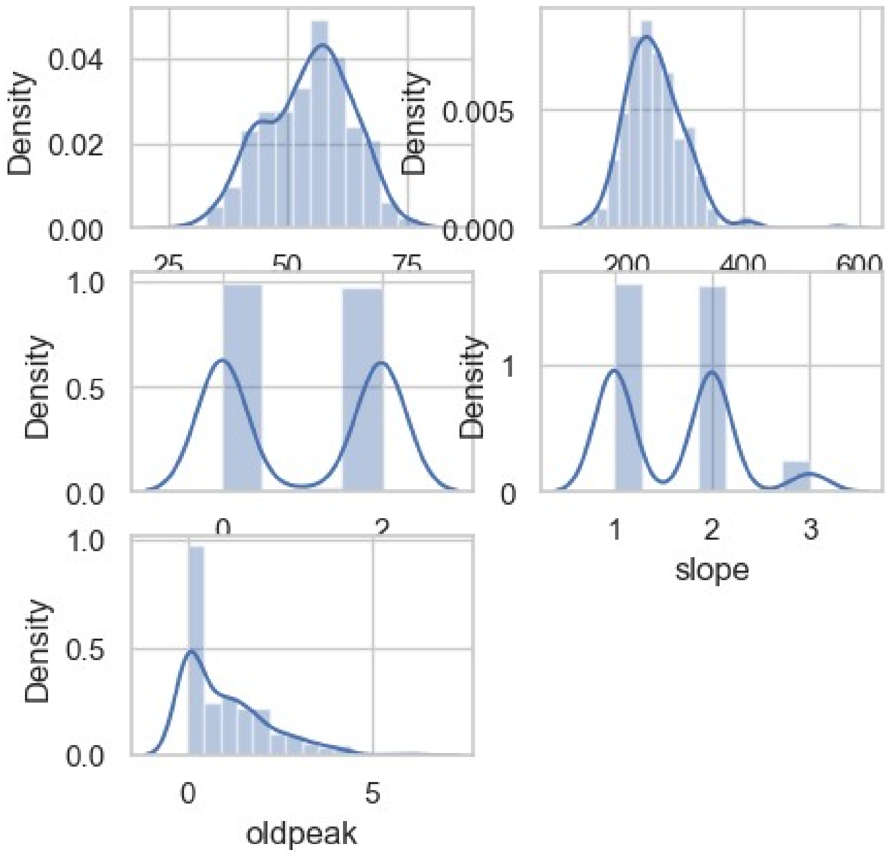



```
sns.distplot(data['chol'])
```

```
<AxesSubplot:xlabel='chol', ylabel='Density'>
```

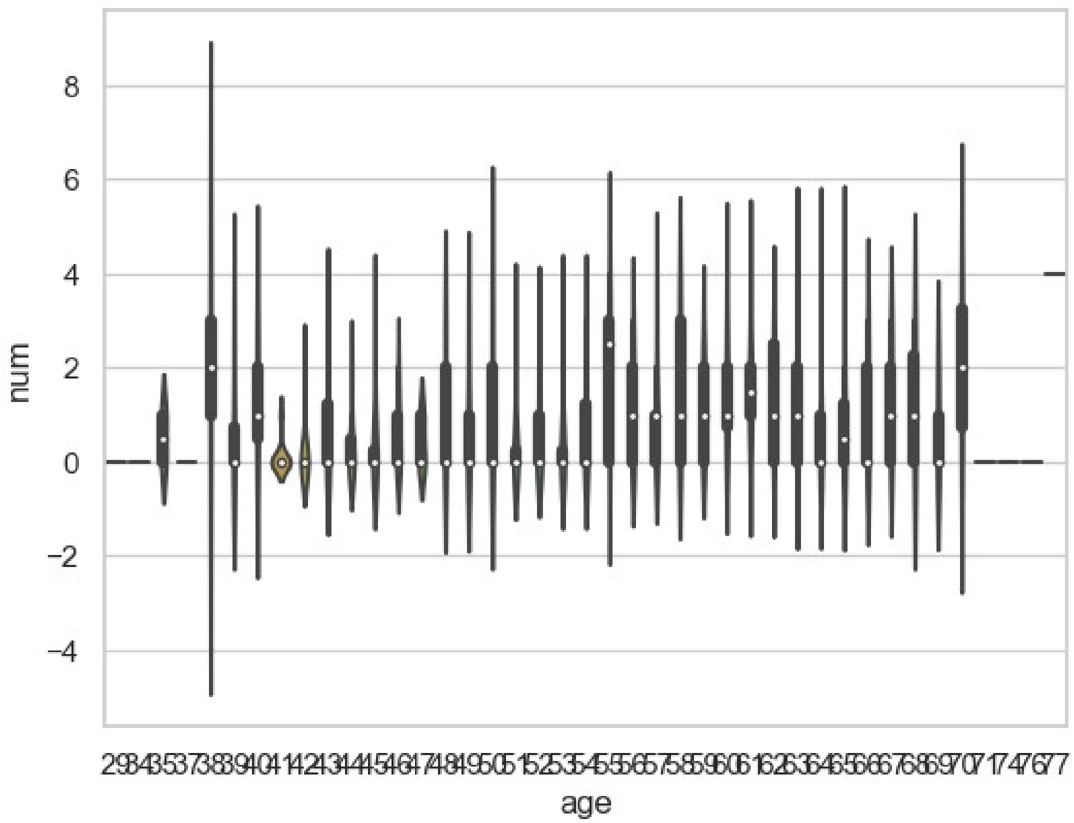



```
features = ['age','chol','restecg','slope', 'oldpeak']
plt.subplots(figsize=(5,5))

for i, col in enumerate(features):
  plt.subplot(3,2,i+1)
  sns.distplot(data[col])
plt.show()
```



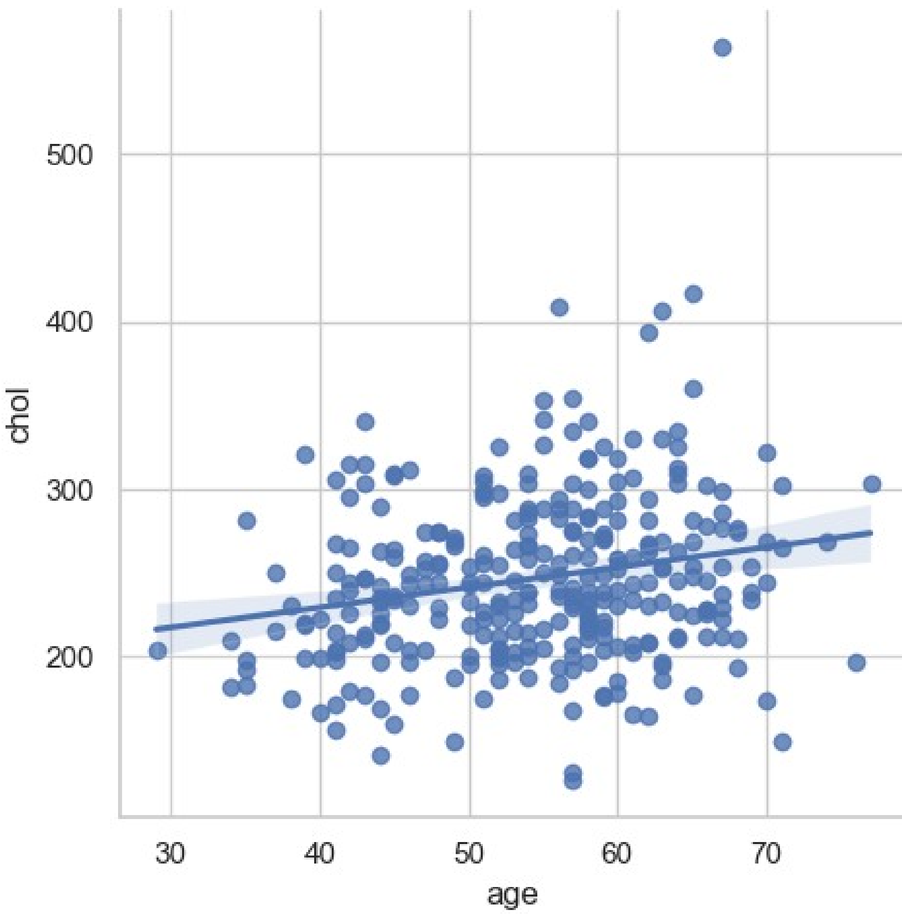

```
sns.violinplot(x='age', y='num', data=data)
```

```
<AxesSubplot:xlabel='age', ylabel='num'>
```



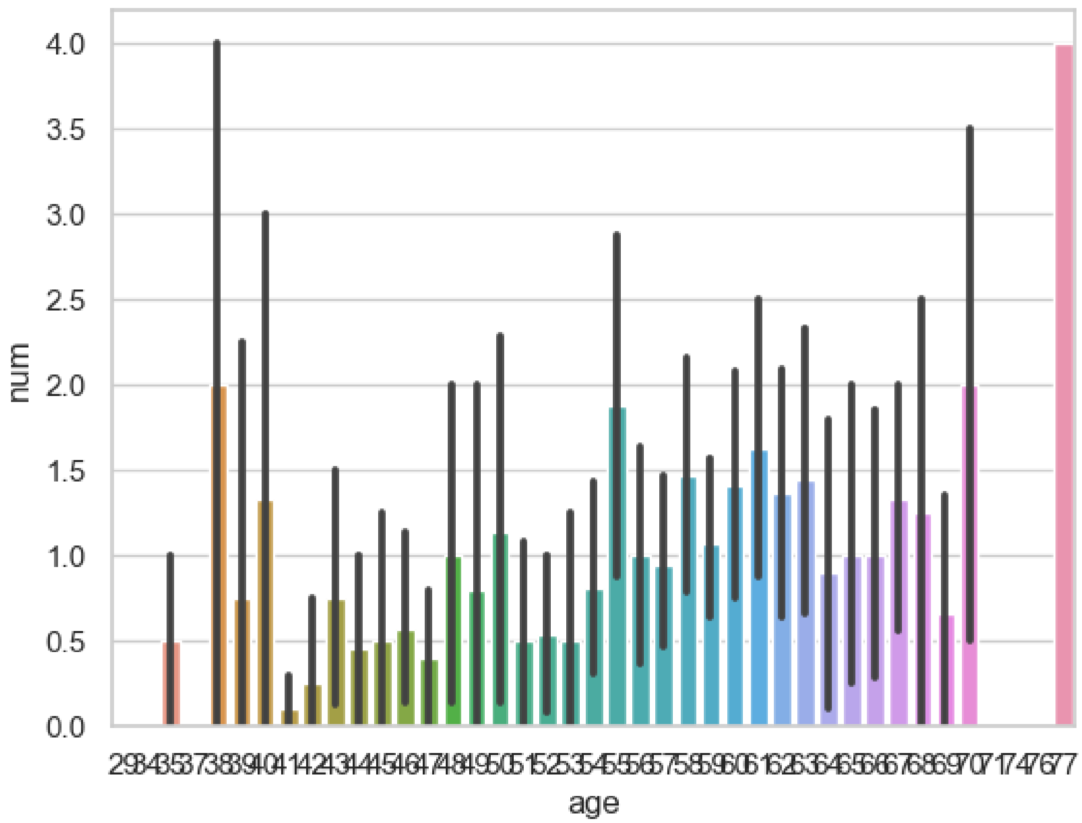

```
sns.lmplot(x='age',y='chol',data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x17845117a00>
```




```
sns.barplot(x='age',y='num',data=data)
```

```
<AxesSubplot:xlabel='age', ylabel='num'>
```




```
data.value_counts()

age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
slope  ca   thal  num
29   1    2   130       204   0    2        202      0      0.0        1
0.0  3.0   0       1
59   1    4   164       176   1    2        90       0      1.0        2
2.0  6.0   3       1
              138       271   0    2        182      0      0.0        1
0.0  3.0   0       1
              135       234   0    0        161      0      0.5        2
0.0  7.0   0       1
              110       239   0    2        142      1      1.2        2
1.0  7.0   2       1

..
51   1    3   110       175   0    0        123      0      0.6        1
0.0  3.0   0       1
              100       222   0    0        143      1      1.2        2
0.0  3.0   0       1
              94        227   0    0        154      1      0.0        1
1.0  7.0   0       1
          1   125       213   0    2        125      1      1.4        1
1.0  3.0   0       1
```

```
77   1    4   125        304   0    2          162       1       0.0       1
3.0  3.0   4       1
Length: 297, dtype: int64
```

```
data['chol'].value_counts()
```

```
204    6
197    6
234    6
269    5
212    5
      ..
340    1
160    1
394    1
184    1
131    1
Name: chol, Length: 152, dtype: int64
```

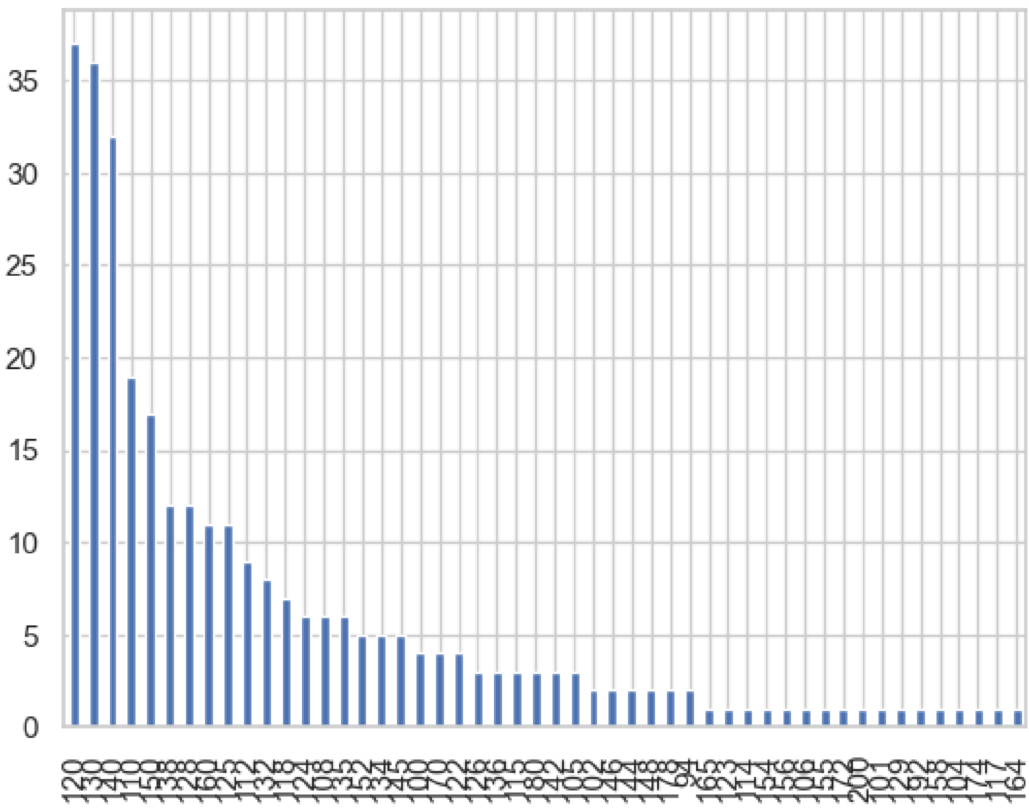
```
data['trestbps'].value_counts()
```

```
120    37
130    36
140    32
110    19
150    17
138    12
128    12
160    11
125    11
112     9
132     8
118     7
124     6
108     6
135     6
152     5
134     5
145     5
100     4
170     4
122     4
126     3
136     3
115     3
180     3
142     3
105     3
102     2
146     2
```

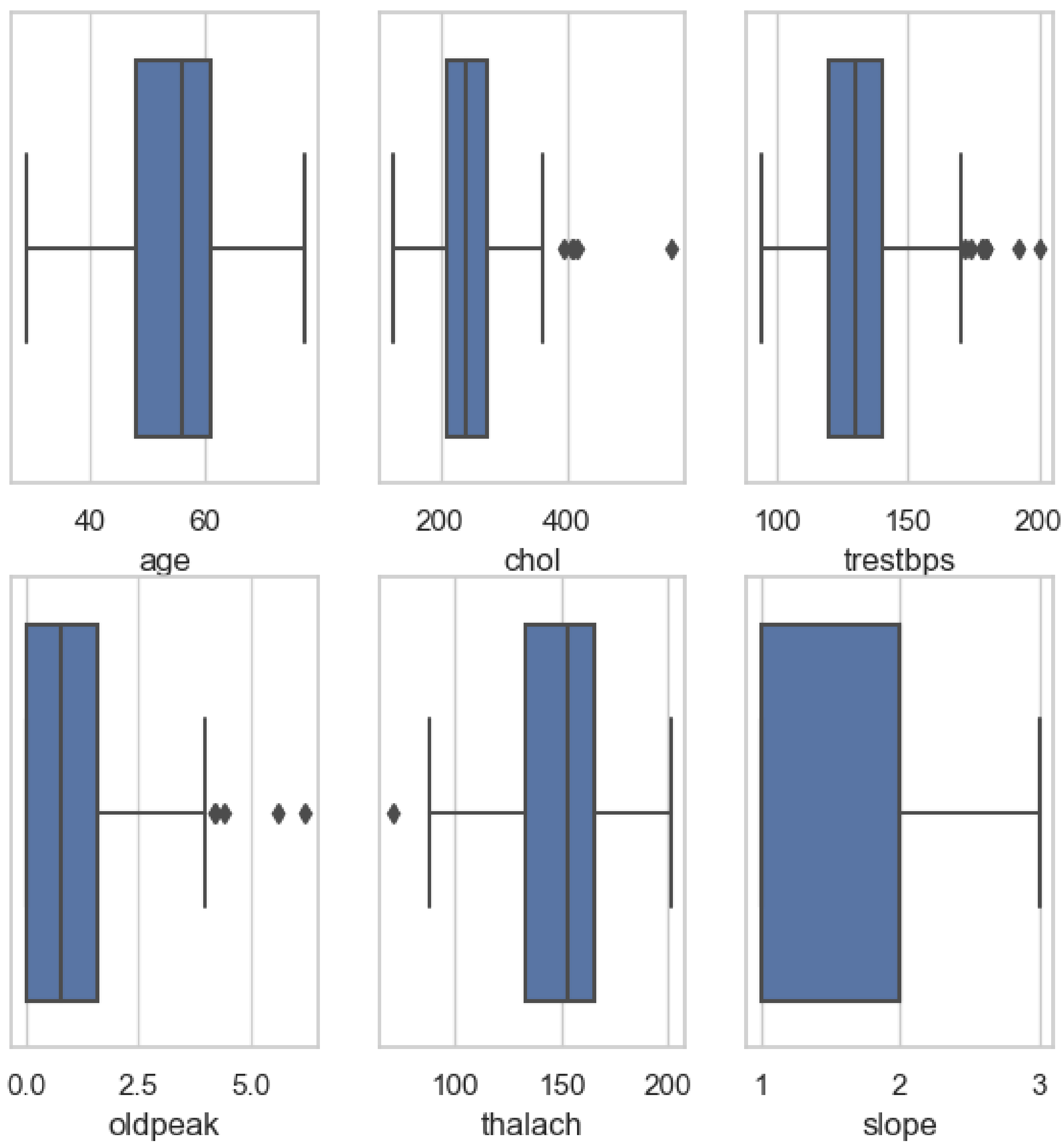
```
144    2
148    2
178    2
94     2
165    1
123    1
114    1
154    1
156    1
106    1
155    1
172    1
200    1
101    1
129    1
192    1
158    1
104    1
174    1
117    1
164    1
Name: trestbps, dtype: int64
```

```
data['trestbps'].value_counts().plot(kind='bar')
```

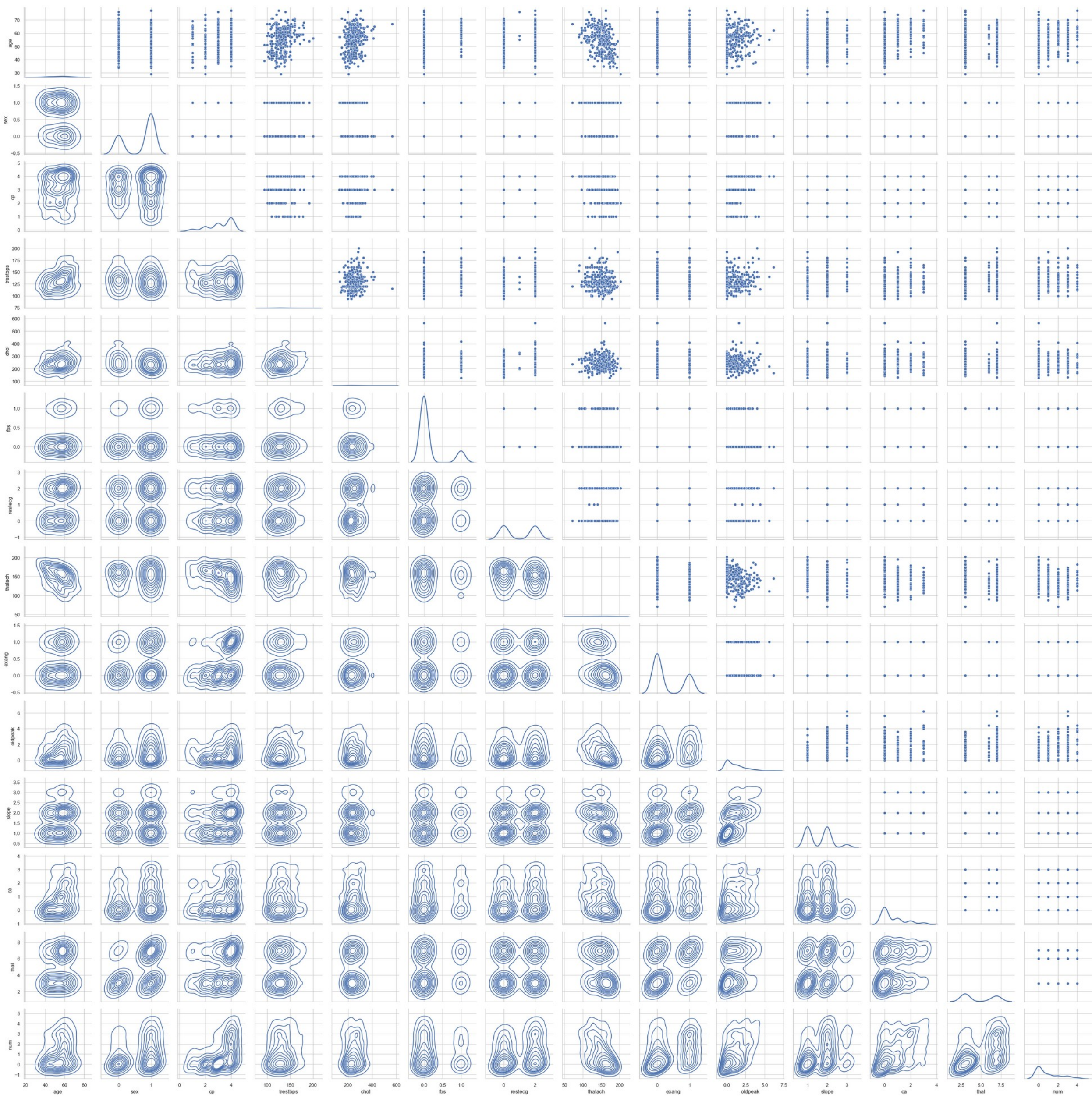
```
<AxesSubplot:>
```




```python
features = ['age','chol','trestbps','oldpeak','thalach','slope']

plt.subplots(figsize=(7,7))

for i, col in enumerate(features):
  plt.subplot(2,3,i+1)
  sns.boxplot(data[col])
plt.show()
```




```
graph=sns.PairGrid(data)

graph=graph.map_upper(sns.scatterplot)

graph=graph.map_lower(sns.kdeplot)

graph=graph.map_diag(sns.kdeplot,lw=2)

plt.show()
```



```python
import plotly.express as px
fig=px.scatter_3d(data,x='age',y='chol',z='num',
                  size='slope',
                  color='fbs')
fig.show()
```


```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":
[{"hovertemplate":"age=%{x}<br>chol=%{y}<br>num=%{z}<br>slope=%
{marker.size}<br>fbs=%{marker.color}<extra></
extra>","legendgroup":"","marker":{"color":
[1,0,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0
,0,0,0,0,1,0,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
```

,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0
,0,0,0,0,0,0,1,0,1,0,0,1,0,1,0,1,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,1
,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0
,0,1,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0
,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1
,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0],"coloraxis":"coloraxis
","size":
[3,2,2,3,1,1,3,1,2,3,2,2,2,1,1,1,3,1,1,1,2,1,2,1,2,2,1,3,1,2,1,1,1,2,1
,1,2,2,2,2,2,1,1,1,1,2,1,2,1,3,1,1,1,1,1,2,2,1,3,1,2,3,2,1,2,2,2,1,3,2
,1,2,2,1,1,1,2,1,2,1,2,2,1,2,1,1,1,1,1,1,2,3,2,2,1,1,2,2,1,1,1,1,1,1,2
,1,1,2,2,2,2,2,2,2,2,2,1,1,1,2,1,2,2,3,2,2,3,2,1,1,2,1,1,1,2,2,3,2,2,2
,1,2,1,2,2,1,2,1,1,1,2,2,2,2,3,2,1,1,2,1,1,1,2,2,1,1,1,1,1,2,2,1,2,2,2
,2,1,2,1,1,2,2,1,3,1,1,3,2,1,2,1,2,2,2,2,2,2,2,1,1,1,1,1,1,1,2,2,2,1,2
,1,2,2,2,1,2,1,2,2,1,1,1,1,2,2,1,1,1,2,1,2,2,1,1,1,2,3,1,2,1,1,1,1,2,1
,2,1,2,1,1,2,2,2,1,2,2,1,2,1,1,2,1,1,1,2,2,2,2,1,1,1,1,2,2,1,2,2,2,1,2
,2,1,2,1,1,3,2,2,1,3,2,1,3,1,2,1,2,2,2,2,2,2,1],"sizemode":"area","siz
eref":7.5e-
3,"symbol":"circle"},"mode":"markers","name":"","scene":"scene","showl
egend":false,"type":"scatter3d","x":
[63,67,67,37,41,56,62,57,63,53,57,56,56,44,52,57,48,54,48,49,64,58,58,
58,60,50,58,66,43,40,69,60,64,59,44,42,43,57,55,61,65,40,71,59,61,58,5
1,50,65,53,41,65,44,44,60,54,50,41,54,51,51,46,58,54,54,60,60,54,59,46
,65,67,62,65,44,65,60,51,48,58,45,53,39,68,52,44,47,53,53,51,66,62,62,
44,63,52,59,60,52,48,45,34,57,71,49,54,59,57,61,39,61,56,52,43,62,41,5
8,35,63,65,48,63,51,55,65,45,56,54,44,62,54,51,29,51,43,55,70,62,35,51
,59,59,52,64,58,47,57,41,45,60,52,42,67,55,64,70,51,58,60,68,46,77,54,
58,48,57,52,54,35,45,70,53,59,62,64,57,52,56,43,53,48,56,42,59,60,63,4
2,66,54,69,50,51,43,62,68,67,69,45,50,59,50,64,57,64,43,45,58,50,55,62
,37,38,41,66,52,56,46,46,64,59,41,54,39,53,63,34,47,67,54,66,52,55,49,
74,54,54,56,46,49,42,41,41,49,61,60,67,58,47,52,62,57,58,64,51,43,42,6
7,76,70,57,44,58,60,44,61,42,52,59,40,42,61,66,46,71,59,64,66,39,57,58
,57,47,55,35,61,58,58,58,56,56,67,55,44,63,63,41,59,57,45,68,57,57,38]
,"y":
[233,286,229,250,204,236,268,354,254,203,192,294,256,263,199,168,229,2
39,275,266,211,283,284,224,206,219,340,226,247,167,239,230,335,234,233
,226,177,276,353,243,225,199,302,212,330,230,175,243,417,197,198,177,2
90,219,253,266,233,172,273,213,305,177,216,304,188,282,185,232,326,231
,269,254,267,248,197,360,258,308,245,270,208,264,321,274,325,235,257,2
16,234,256,302,164,231,141,252,255,239,258,201,222,260,182,303,265,188
,309,177,229,260,219,307,249,186,341,263,203,211,183,330,254,256,407,2
22,217,282,234,288,239,220,209,258,227,204,261,213,250,174,281,198,245
,221,288,205,309,240,243,289,250,308,318,298,265,564,289,246,322,299,3
00,293,277,197,304,214,248,255,207,223,288,282,160,269,226,249,394,212
,274,233,184,315,246,274,409,244,270,305,195,240,246,283,254,196,298,2
47,294,211,299,234,236,244,273,254,325,126,313,211,309,259,200,262,244
,215,231,214,228,230,193,204,243,303,271,268,267,199,282,269,210,204,2
77,206,212,196,327,149,269,201,286,283,249,271,295,235,306,269,234,178
,237,234,275,212,208,201,218,263,295,303,209,223,197,245,261,242,319,2

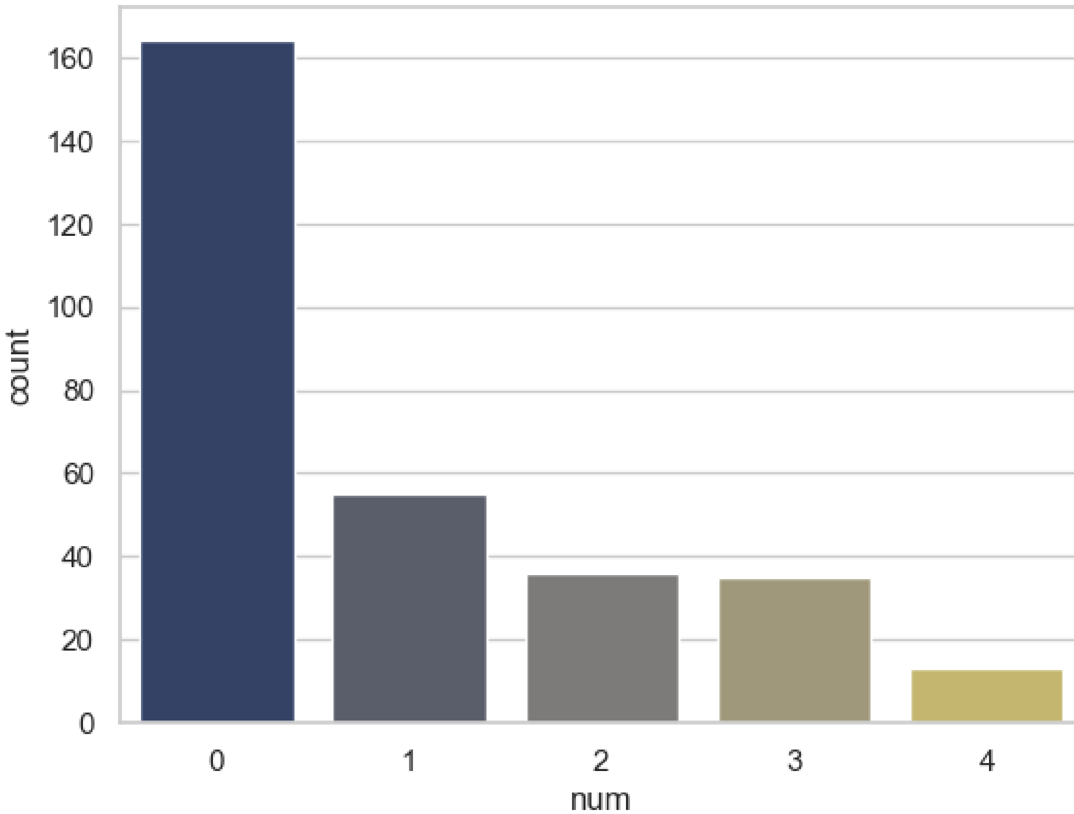
40,226,166,315,204,218,223,180,207,228,311,149,204,227,278,220,232,197
,335,253,205,192,203,318,225,220,221,240,212,342,169,187,197,157,176,2
41,264,193,131,236,175],"z":
[0,2,1,0,0,0,3,0,2,1,0,0,2,0,0,0,1,0,0,0,0,0,1,3,4,0,0,0,0,3,0,2,1,0,0
,0,3,1,3,0,4,0,0,0,1,4,0,4,0,0,0,0,2,0,1,1,1,1,0,0,2,0,1,0,2,2,1,0,2,1
,0,3,1,1,1,0,1,0,0,3,0,0,0,3,0,0,0,0,0,0,0,3,0,0,0,1,2,3,0,0,0,0,0,0,3
,0,2,1,2,3,1,1,0,2,2,0,0,0,3,2,3,4,0,3,1,0,3,3,0,0,0,0,0,0,0,0,4,3,1,0
,0,1,0,1,0,1,4,0,0,0,0,0,0,4,3,1,1,1,2,0,0,4,0,0,0,0,0,0,1,0,3,0,1,0,4
,1,0,1,0,0,3,2,0,0,1,0,0,2,1,2,0,3,1,2,0,3,0,0,0,1,0,0,0,0,0,3,3,3,0,1
,0,4,0,3,1,0,0,0,0,0,0,0,0,3,1,0,0,0,3,2,0,2,1,0,0,3,2,1,0,0,0,0,0,2,0
,2,2,1,3,0,0,1,0,0,0,0,0,0,0,1,0,3,0,0,4,2,2,2,1,0,1,0,2,0,1,0,0,0,1,0
,2,0,3,0,2,4,2,0,0,0,1,0,2,2,1,0,3,1,1,2,3,1,0]}],"layout":
{"coloraxis":{"colorbar":{"title":{"text":"fbs"}},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"legend":
{"itemsizing":"constant","tracegroupgap":0},"margin":{"t":60},"scene":
{"domain":{"x":[0,1],"y":[0,1]},"xaxis":{"title":
{"text":"age"}},"yaxis":{"title":{"text":"chol"}},"zaxis":{"title":
{"text":"num"}}},"template":{"data":{"bar":[{"error_x":
{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":

```
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
```

```
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}}}}
```

```
sns.set_style('whitegrid')
sns.countplot(x='num',data=data,palette='cividis')
```

```
<AxesSubplot:xlabel='num', ylabel='count'>
```




```
fig=px.pie(data,values='age',names='fbs',title='target')
fig.show()
```


```
{"config":{"plotlyServerURL":"https://plot.ly"},"data":[{"domain":
{"x":[0,1],"y":[0,1]},"hovertemplate":"fbs=%{label}<br>age=%
{value}<extra></extra>","labels":
[1,0,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0
,0,0,0,0,1,0,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0
,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0
,0,0,0,0,0,0,1,0,1,0,0,1,0,1,0,1,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1
,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,0
,0,1,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,1,1,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0
,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1
,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0],"legendgroup":"","name
":"","showlegend":true,"type":"pie","values":
[63,67,67,37,41,56,62,57,63,53,57,56,56,44,52,57,48,54,48,49,64,58,58,
```

```
58,60,50,58,66,43,40,69,60,64,59,44,42,43,57,55,61,65,40,71,59,61,58,5
1,50,65,53,41,65,44,44,60,54,50,41,54,51,51,46,58,54,54,60,60,54,59,46
,65,67,62,65,44,65,60,51,48,58,45,53,39,68,52,44,47,53,53,51,66,62,62,
44,63,52,59,60,52,48,45,34,57,71,49,54,59,57,61,39,61,56,52,43,62,41,5
8,35,63,65,48,63,51,55,65,45,56,54,44,62,54,51,29,51,43,55,70,62,35,51
,59,59,52,64,58,47,57,41,45,60,52,42,67,55,64,70,51,58,60,68,46,77,54,
58,48,57,52,54,35,45,70,53,59,62,64,57,52,56,43,53,48,56,42,59,60,63,4
2,66,54,69,50,51,43,62,68,67,69,45,50,59,50,64,57,64,43,45,58,50,55,62
,37,38,41,66,52,56,46,46,64,59,41,54,39,53,63,34,47,67,54,66,52,55,49,
74,54,54,56,46,49,42,41,41,49,61,60,67,58,47,52,62,57,58,64,51,43,42,6
7,76,70,57,44,58,60,44,61,42,52,59,40,42,61,66,46,71,59,64,66,39,57,58
,57,47,55,35,61,58,58,58,56,56,67,55,44,63,63,41,59,57,45,68,57,57,38]
}],"layout":{"legend":{"tracegroupgap":0},"template":{"data":{"bar":
[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
```

```
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
```

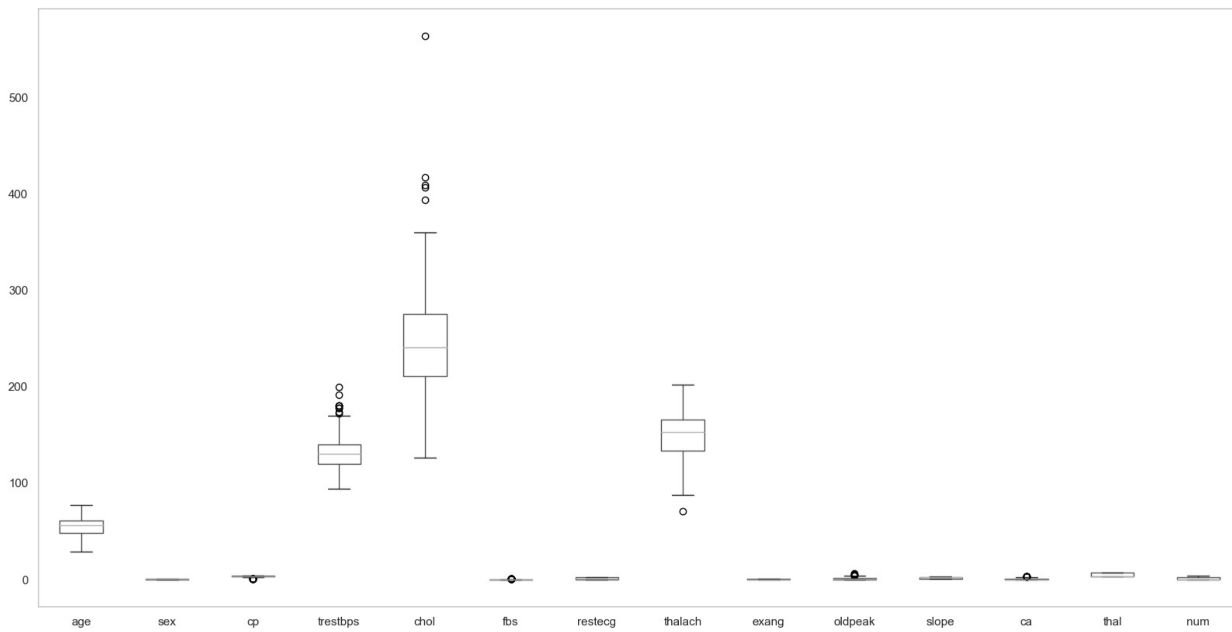
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"title":
{"text":"target"}}}

```
data.nunique().sort_values()
```

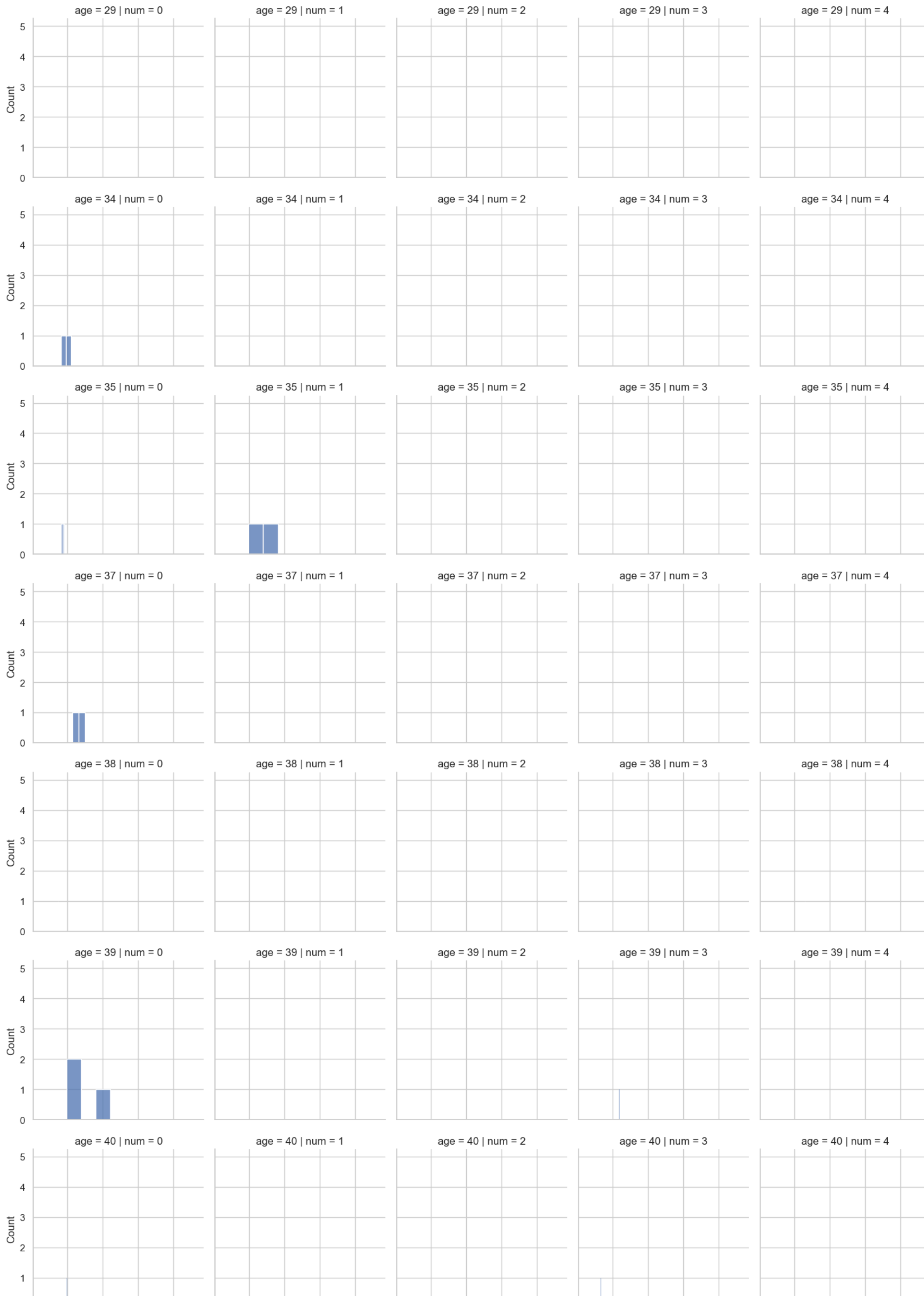
ColName	
sex	2
fbs	2
exang	2

```
restecg       3
slope         3
thal          3
cp            4
ca            4
num           5
oldpeak      40
age          41
trestbps     50
thalach      91
chol        152
dtype: int64
```

```
plt.figure(figsize=(20,10))
data.boxplot(grid=False)
plt.show()
```




```
a=sns.FacetGrid(data,row='age',col='num')
a.map(sns.histplot,'chol')
plt.show()
```


age = 29 | num = 0
age = 29 | num = 1
age = 29 | num = 2
age = 29 | num = 3
age = 29 | num = 4
age = 34 | num = 0
age = 34 | num = 1
age = 34 | num = 2
age = 34 | num = 3
age = 34 | num = 4
age = 35 | num = 0
age = 35 | num = 1
age = 35 | num = 2
age = 35 | num = 3
age = 35 | num = 4
age = 37 | num = 0
age = 37 | num = 1
age = 37 | num = 2
age = 37 | num = 3
age = 37 | num = 4
age = 38 | num = 0
age = 38 | num = 1
age = 38 | num = 2
age = 38 | num = 3
age = 38 | num = 4
age = 39 | num = 0
age = 39 | num = 1
age = 39 | num = 2
age = 39 | num = 3
age = 39 | num = 4
age = 40 | num = 0
age = 40 | num = 1
age = 40 | num = 2
age = 40 | num = 3
age = 40 | num = 4
Count
0
1
2
3
4
5

```
data.replace('?',' ',inplace=True)
data
```

```
ColName  age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  
oldpeak  \
0         63    1   1       145   233    1        2      150      0   
2.3   
1         67    1   4       160   286    0        2      108      1   
1.5   
2         67    1   4       120   229    0        2      129      1   
2.6   
3         37    1   3       130   250    0        0      187      0   
3.5   
4         41    0   2       130   204    0        2      172      0   
1.4   
..       ...  ...  ..       ...   ...  ...      ...      ...    ...   
...   
298       45    1   1       110   264    0        0      132      0   
1.2   
299       68    1   4       144   193    1        0      141      0   
3.4   
300       57    1   4       130   131    0        0      115      1   
1.2   
301       57    0   2       130   236    0        2      174      0   
0.0   
302       38    1   3       138   175    0        0      173      0   
0.0   

ColName  slope   ca  thal  num  
0            3  0.0   6.0    0  
1            2  3.0   3.0    2  
2            2  2.0   7.0    1  
3            3  0.0   3.0    0  
4            1  0.0   3.0    0  
..         ...  ...   ...  ...  
298          2  0.0   7.0    1  
299          2  2.0   7.0    2  
300          2  1.0   7.0    3  
301          2  1.0   3.0    1  
302          1  NaN   3.0    0  

[303 rows x 14 columns]
```

```
data['ca'].replace('',np.nan, inplace=True)
data['thal'].replace('',np.nan, inplace=True)
data.dropna(subset=['ca'],inplace=True)
data.dropna(subset=['thal'],inplace=True)
```

TRAINING AND TESTING DATA

```
cat_val=[]
count_val=[]

for col in data.columns:
    if data[col].nunique()<=10:
        cat_val.append(col)
    else:
        count_val.append(col)

cat_val

['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'num']

count_val

['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

data=pd.get_dummies(data,columns=['cp','fbs','restecg','exang','slope'
,'ca','thal'])
data.head()

   age  sex  trestbps  chol  thalach  oldpeak  num  cp_1  cp_2
cp_3  ...  \
0   63    1       145   233      150      2.3    0     1     0
0  ...
1   67    1       160   286      108      1.5    2     0     0
0  ...
2   67    1       120   229      129      2.6    1     0     0
0  ...
3   37    1       130   250      187      3.5    0     0     0
1  ...
4   41    0       130   204      172      1.4    0     0     1
0  ...

   slope_1  slope_2  slope_3  ca_0.0  ca_1.0  ca_2.0  ca_3.0  thal_3.0
\
0        0        0        1       1       0       0       0        0

1        0        1        0       0       0       0       1        1

2        0        1        0       0       0       1       0        0

3        0        0        1       1       0       0       0        1

4        1        0        0       1       0       0       0        1


   thal_6.0  thal_7.0
0         1         0
```

```
1         0         0
2         0         1
3         0         0
4         0         0

[5 rows x 28 columns]
```

```
x=data.drop(['num'],axis=1)
y=data['num']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,rando
m_state=1)

ss=StandardScaler()
x_train=ss.fit_transform(x_train)
x_test=ss.transform(x_test)
```

Random Forest

```
clfr=RandomForestClassifier(n_estimators=10,criterion='entropy',random
_state=0)

clfr.fit(x_train,y_train)

RandomForestClassifier(criterion='entropy', n_estimators=10,
random_state=0)

clfr1=RandomForestClassifier(n_estimators=10,criterion='gini',random_s
tate=0)

clfr1.fit(x_train,y_train)

RandomForestClassifier(n_estimators=10, random_state=0)

ypre1=clfr.predict(x_test)# entropy ypre calculation

ypre2=clfr1.predict(x_test)# gini ypre calculation

print('entropy Accuracy Score:')
accuracy_score(y_test,ypre1)*100

entropy Accuracy Score:

55.00000000000001

print('gini Accuracy Score:')
accuracy_score(y_test,ypre2)*100

gini Accuracy Score:
```

```
53.333333333333336
```

```python
print('entropy - confusion matrix\n--------------------\n')
print(confusion_matrix(y_test,ypre1))
print('gini - confusion matrix\n--------------------\n')
print(confusion_matrix(y_test,ypre2))
```

```
entropy - confusion matrix
--------------------

[[27  3  0  0  0]
 [ 3  5  1  1  0]
 [ 5  3  0  3  0]
 [ 2  3  1  1  1]
 [ 0  0  0  1  0]]
gini - confusion matrix
--------------------

[[26  3  0  1  0]
 [ 5  3  1  0  1]
 [ 6  2  0  3  0]
 [ 2  1  1  3  1]
 [ 1  0  0  0  0]]
```

```python
print('entropy result\n--------------')
print(classification_report(y_test,ypre1))
print('gini index result\n----------------------')
print(classification_report(y_test,ypre2))
```

```
entropy result
--------------
              precision    recall  f1-score   support

           0       0.73      0.90      0.81        30
           1       0.36      0.50      0.42        10
           2       0.00      0.00      0.00        11
           3       0.17      0.12      0.14         8
           4       0.00      0.00      0.00         1

    accuracy                           0.55        60
   macro avg       0.25      0.30      0.27        60
weighted avg       0.45      0.55      0.49        60

gini index result
----------------------
              precision    recall  f1-score   support

           0       0.65      0.87      0.74        30
           1       0.33      0.30      0.32        10
           2       0.00      0.00      0.00        11
           3       0.43      0.38      0.40         8
```

```
              4       0.00      0.00      0.00         1

    accuracy                            0.53        60
   macro avg       0.28      0.31      0.29        60
weighted avg       0.44      0.53      0.48        60
```

Decision Tree

```
dtree = DecisionTreeClassifier(max_depth=6, random_state=1)

dtree.fit(x_train,y_train)

DecisionTreeClassifier(max_depth=6, random_state=1)

y_pre3=dtree.predict(x_test)
from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score,mean_squared_err
or
print(classification_report(y_test,y_pre3))

              precision    recall  f1-score   support

           0       0.76      0.83      0.79        30
           1       0.18      0.20      0.19        10
           2       0.33      0.27      0.30        11
           3       0.00      0.00      0.00         8
           4       0.00      0.00      0.00         1

    accuracy                           0.50        60
   macro avg       0.25      0.26      0.26        60
weighted avg       0.47      0.50      0.48        60


print(confusion_matrix(y_test,y_pre3))
print("Training Score: ",dtree.score(x_train,y_train)*100)

[[25  3  0  2  0]
 [ 3  2  3  1  1]
 [ 4  3  3  1  0]
 [ 1  2  3  0  2]
 [ 0  1  0  0  0]]
Training Score:  82.70042194092827

print(accuracy_score(y_test,y_pre3)*100)

50.0

data = pd.DataFrame({'Actual': y_test, 'Predicted': y_pre3})
data.head()
```

```
     Actual  Predicted
139       0          0
236       2          1
51        0          1
295       0          0
245       2          0
```

Logistic Regression

```
reg = LogisticRegression()
reg.fit(x_train,y_train)

LogisticRegression()

y_pre4=reg.predict(x_test)
y_pre4

array([0, 3, 0, 0, 1, 1, 0, 4, 0, 2, 3, 2, 0, 1, 1, 1, 3, 0, 0, 0, 1,
0,
       0, 0, 0, 0, 0, 3, 1, 0, 0, 1, 0, 4, 2, 0, 0, 0, 0, 4, 0, 0, 0,
1,
       0, 0, 0, 1, 3, 4, 0, 1, 3, 1, 4, 3, 4, 4, 0, 1], dtype=int64)

print(classification_report(y_test,y_pre4))
print(confusion_matrix(y_test,y_pre4))
print("Training Score: ",reg.score(x_train,y_train)*100)

              precision    recall  f1-score   support

           0       0.80      0.80      0.80        30
           1       0.23      0.30      0.26        10
           2       0.00      0.00      0.00        11
           3       0.14      0.12      0.13         8
           4       0.00      0.00      0.00         1

    accuracy                           0.47        60
   macro avg       0.23      0.25      0.24        60
weighted avg       0.46      0.47      0.46        60

[[24  3  0  1  2]
 [ 4  3  0  1  2]
 [ 2  4  0  4  1]
 [ 0  3  2  1  2]
 [ 0  0  1  0  0]]
Training Score:  73.83966244725738

data = pd.DataFrame({'Actual': y_test, 'Predicted': y_pre4})
data.head()
```

```
     Actual  Predicted
139       0          0
236       2          3
51        0          0
295       0          0
245       2          1
```

```
print(accuracy_score(y_test,y_pre4)*100)
```

```
46.666666666666664
```

```
from sklearn.model_selection import GridSearchCV
param = {
         'penalty':['l1','l2'],
         'C':[0.001, 0.01, 0.1, 1, 10, 20,100, 1000]
}
lr= LogisticRegression(penalty='l1')
cv=GridSearchCV(reg,param,cv=5,n_jobs=-1)
cv.fit(x_train,y_train)
cv.predict(x_test)
```

```
array([0, 3, 0, 0, 0, 1, 0, 4, 0, 2, 3, 2, 0, 1, 1, 0, 3, 0, 0, 0, 1,
0,
       0, 0, 0, 0, 0, 3, 1, 0, 0, 0, 0, 4, 2, 0, 0, 0, 0, 4, 0, 0, 0,
1,
       0, 0, 0, 1, 0, 4, 0, 1, 3, 1, 4, 3, 4, 4, 0, 1], dtype=int64)
```

```
print("Best CV score", cv.best_score_*100)
```

```
Best CV score 59.9290780141844
```

KNN

```
knn=KNeighborsClassifier(n_neighbors=7)
```

```
knn.fit(x_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=7)
```

```
y_pre5=knn.predict(x_test)
```

```
data = pd.DataFrame({'Actual': y_test, 'Predicted': y_pre5})
data.head()
```

```
     Actual  Predicted
139       0          0
236       2          2
51        0          0
```

```
295       0          0
245       2          0
```

```
from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score,mean_squared_err
or,r2_score
print(classification_report(y_test,y_pre5))
```

```
              precision    recall  f1-score   support

           0       0.64      0.93      0.76        30
           1       0.25      0.20      0.22        10
           2       0.25      0.09      0.13        11
           3       0.00      0.00      0.00         8
           4       0.00      0.00      0.00         1

    accuracy                           0.52        60
   macro avg       0.23      0.24      0.22        60
weighted avg       0.41      0.52      0.44        60
```

```
print(confusion_matrix(y_test,y_pre5))
```

```
[[28  2  0  0  0]
 [ 6  2  0  1  1]
 [ 6  3  1  1  0]
 [ 3  1  3  0  1]
 [ 1  0  0  0  0]]
```

```
print("Training Score: ",knn.score(x_train,y_train)*100)
print(knn.score(x_test,y_test))
```

```
Training Score:  66.66666666666666
0.5166666666666667
```

```
accuracy_score(y_test,y_pre5)
```

```
0.5166666666666667
```

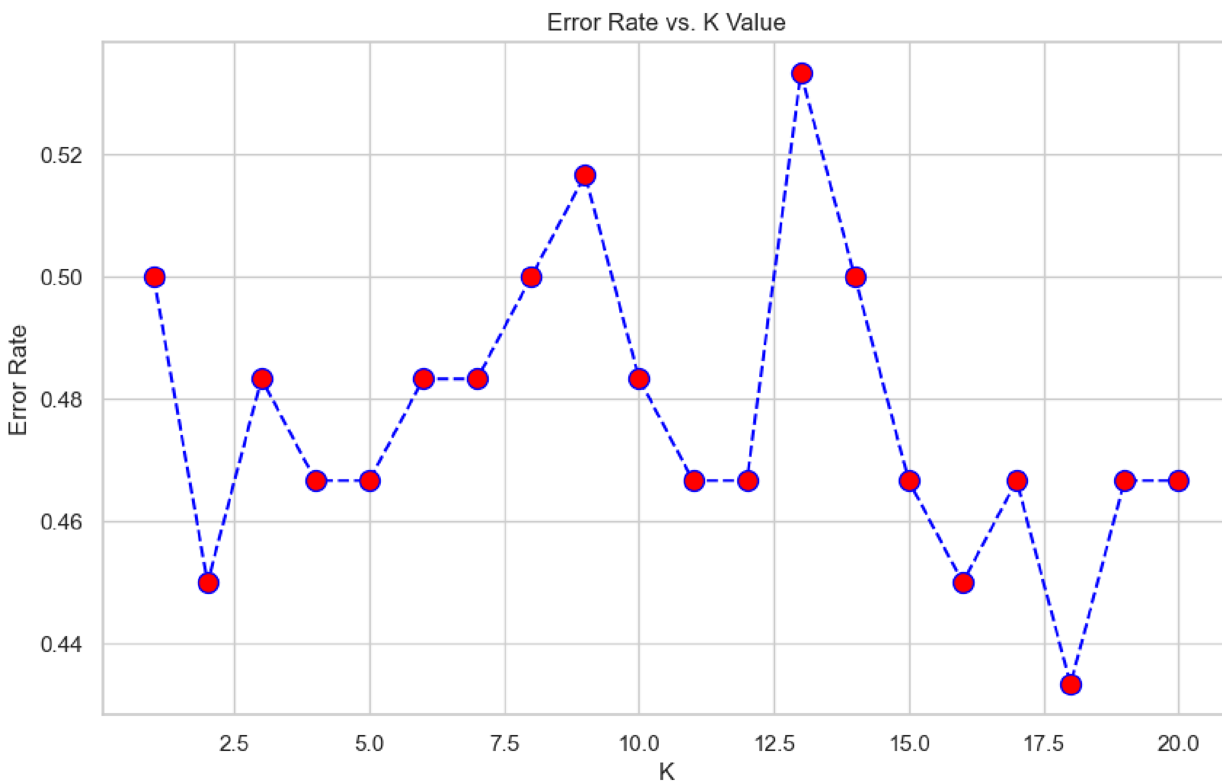
```
t=1-accuracy_score(y_test,y_pre5)
t
```

```
0.48333333333333333
```

```
error_rate = []

for i in range(1,21):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train,y_train)
    pred_i = knn.predict(x_test)
```

```
    t=1-accuracy_score(y_test,pred_i)
    error_rate.append(t)

plt.figure(figsize=(10,6))
plt.plot(range(1,21),error_rate,color='blue', linestyle='dashed',
marker='o',markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

```
Text(0, 0.5, 'Error Rate')
```




```
data=pd.DataFrame({'Models':
['Rf(Entropy)','Rf(Gini)','DT','Logreg','Knn'],
                'Accuracy':
[accuracy_score(y_test,ypre1)*100,accuracy_score(y_test,ypre2)*100,

accuracy_score(y_test,y_pre3)*100,accuracy_score(y_test,y_pre4)*100,
                            accuracy_score(y_test,y_pre5)*100]})
data
```

	Models	Accuracy
0	Rf(Entropy)	55.000000
1	Rf(Gini)	53.333333
2	DT	50.000000

```
3        Logreg  46.666667
4           Knn  51.666667
```

```
sns.barplot(data['Models'],data['Accuracy'])
plt.xticks(rotation=90)
plt.show()
```