DRINKING STATUS

This dataset is collected from National Health Insurance Service in Korea. The purpose of this dataset is to:

- Analysis of body signal
- To predict whether a person is a drinker or not according to collected Features ✅ ❌

The dataset contains 23 following features.

- Sex male, female
- Age round up to 5 years
- Height round up to 5 cm (cm)
- Weight (kg)
- Sight_left eyesight (left)
- Sight_right eyesight (right)
- Hear_left hearing left, 1 (normal), 2 (abnormal)
- Hear_right hearing right, 1 (normal), 2 (abnormal)
- SBP Systolic blood pressure (mmHg)
- DBP Diastolic blood pressure (mmHg)
- BLDS Blood Sugar Level or Fasting Blood Glucose (mg/dL)
- Tot_chole total cholesterol (mg/dL)
- HDL_chole High-Density Lipoprotein cholesterol (mg/dL)
- LDL_chole Low-Density Lipoprotein cholesterol (mg/dL)

- LDL_chole Low-Density Lipoprotein cholesterol (mg/dL)
- Triglyceride triglyceride (mg/dL)
- Hemoglobin hemoglobin (g/dL)
- Urine_protein protein in urine
- Serum_creatinine serum (blood) creatinine (mg/dL)
- SGOT_AST Serum Glutamate Oxaloacetate Transaminase Aspartate Transaminase (IU/L)
- SGOT_ALT Serum Glutamate Oxaloacetate Transaminase Alanine Transaminase (IU/L)
- Gamma_GTP γ-Glutamyl Transpeptidase (IU/L)
- DRK_YN Drinker or Not

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

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

Importing neccessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/drive/MyDrive/driking_dataset_Ver01.csv')
df=df.drop(['SMK_stat_type_cd'], axis = 1)
df
```

	sex	age	height	weight	waistline	sight_left	sight_right	hear_left	hear_
0	Male	35	170.0	75.0	90.0	1.0	1.0	1	
1	Male	30	180.0	80.0	89.0	0.9	1.2	1	
2	Male	40	165.0	75.0	91.0	1.2	1.5	1	
3	Male	50	175.0	80.0	91.0	1.5	1.2	1	
4	Male	50	165.0	60.0	80.0	1.0	1.2	1	
...	...	...	...	...	...	...	...	...	
23994	Female	50	155.0	55.0	71.2	1.0	1.2	1	
23995	Male	55	165.0	65.0	84.0	1.0	0.8	1	
23996	Male	40	170.0	75.0	90.0	1.2	1.2	1	
23997	Female	30	150.0	55.0	76.0	1.2	1.0	1	
23998	Male	50	165.0	70.0	94.0	0.7	0.6	1	

23999 rows × 23 columns

```
1 df.head()
```

	sex	age	height	weight	waistline	sight_left	sight_right	hear_left	hear_right
0	Male	35	170.0	75.0	90.0	1.0	1.0	1	1
1	Male	30	180.0	80.0	89.0	0.9	1.2	1	1
2	Male	40	165.0	75.0	91.0	1.2	1.5	1	1
3	Male	50	175.0	80.0	91.0	1.5	1.2	1	1
4	Male	50	165.0	60.0	80.0	1.0	1.2	1	1

5 rows × 23 columns

```
1 df.tail()
```

	sex	age	height	weight	waistline	sight_left	sight_right	hear_left	hear_
23994	Female	50	155.0	55.0	71.2	1.0	1.2	1	
23995	Male	55	165.0	65.0	84.0	1.0	0.8	1	
23996	Male	40	170.0	75.0	90.0	1.2	1.2	1	
23997	Female	30	150.0	55.0	76.0	1.2	1.0	1	
23998	Male	50	165.0	70.0	94.0	0.7	0.6	1	

5 rows × 23 columns

```
1 df.describe()
```

	age	height	weight	waistline	sight_left	sight_right
count	23999.000000	23936.000000	23971.000000	23999.000000	23998.000000	23999.000000
mean	47.592816	162.290901	63.293146	81.291616	0.981407	0.977557
std	14.165375	9.281372	12.485935	12.729166	0.612472	0.590782
min	20.000000	135.000000	30.000000	35.000000	0.100000	0.100000
25%	35.000000	155.000000	55.000000	74.200000	0.700000	0.700000
50%	45.000000	160.000000	60.000000	81.000000	1.000000	1.000000
75%	60.000000	170.000000	70.000000	87.600000	1.200000	1.200000
max	85.000000	190.000000	130.000000	999.000000	9.900000	9.900000

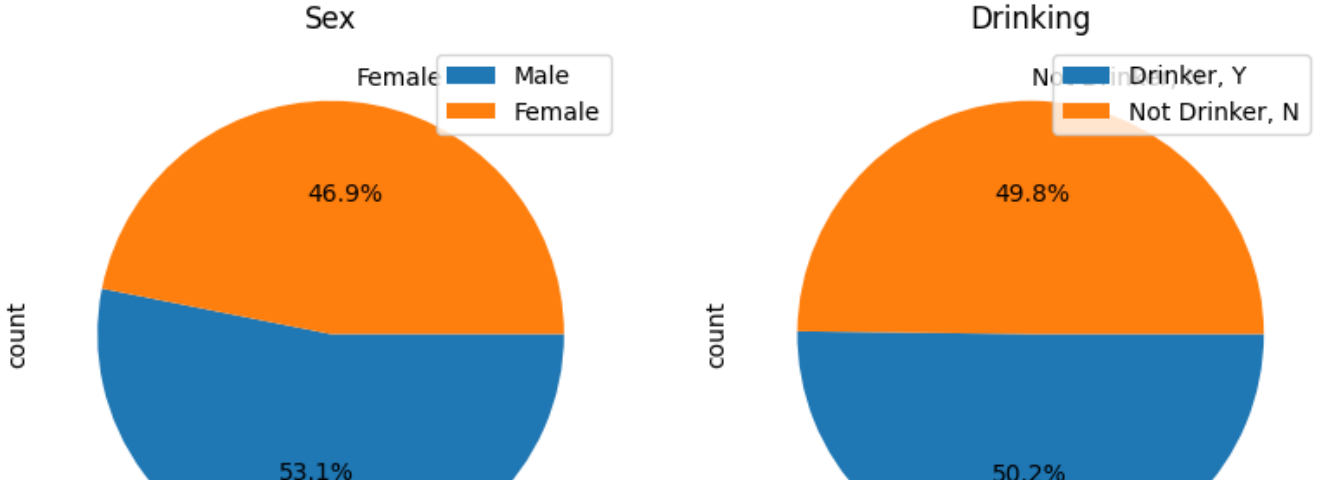
8 rows × 21 columns

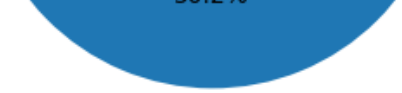
```
1 df.dtypes
```

```
sex                 object
age                  int64
height             float64
weight             float64
waistline          float64
sight_left         float64
sight_right        float64
hear_left            int64
hear_right           int64
SBP                float64
DBP                float64
BLDS               float64
tot_chole          float64
HDL_chole            int64
LDL_chole            int64
triglyceride       float64
hemoglobin         float64
urine_protein        int64
serum_creatinine   float64
SGOT_AST           float64
SGOT_ALT             int64
gamma_GTP            int64
DRK_YN              object
dtype: object
```

Graphical Representation

```
1 plt.figure(figsize=(20,20))
2 plt.subplot(1, 4, 1)
3 label1 = ['Male', 'Female']
4 df['sex'].value_counts().plot.pie(labels=label1,counterclock=False,autopct='%1.1f%%')
5 plt.title('Sex')
6 plt.legend()
7 plt.subplot(1, 4, 2)
8 labels = ['Drinker, Y', 'Not Drinker, N']
9 df['DRK_YN'].value_counts().plot.pie(labels=labels, counterclock=False,autopct='%1.1f%%')
10 plt.title('Drinking')
11 plt.legend()
12 plt.show()
```






```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23999 entries, 0 to 23998
Data columns (total 23 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   sex               23999 non-null  object 
 1   age               23999 non-null  int64  
 2   height            23936 non-null  float64
 3   weight            23971 non-null  float64
 4   waistline         23999 non-null  float64
 5   sight_left        23998 non-null  float64
 6   sight_right       23999 non-null  float64
 7   hear_left         23999 non-null  int64  
 8   hear_right        23999 non-null  int64  
 9   SBP               23956 non-null  float64
 10  DBP               23904 non-null  float64
 11  BLDS              23991 non-null  float64
 12  tot_chole         23990 non-null  float64
 13  HDL_chole         23999 non-null  int64  
 14  LDL_chole         23999 non-null  int64  
 15  triglyceride      23985 non-null  float64
 16  hemoglobin        23999 non-null  float64
 17  urine_protein     23999 non-null  int64  
 18  serum_creatinine  23999 non-null  float64
 19  SGOT_AST          23950 non-null  float64
 20  SGOT_ALT          23999 non-null  int64  
 21  gamma_GTP         23999 non-null  int64  
 22  DRK_YN            23999 non-null  object 
dtypes: float64(13), int64(8), object(2)
memory usage: 4.2+ MB
```

Finding & Filling missing values

```
1 df.isna().sum()
```

```
sex                  0
age                  0
height              63
weight              28
waistline            0
sight_left           1
sight_right          0
hear_left            0
hear_right           0
```

```
SBP                43
DBP                95
BLDS                8
tot_chole           9
HDL_chole           0
LDL_chole           0
triglyceride       14
hemoglobin          0
urine_protein       0
serum_creatinine    0
SGOT_AST           49
SGOT_ALT            0
gamma_GTP           0
DRK_YN              0
dtype: int64
```

```
df['height']=df['height'].fillna(df['height'].mean())
df['weight']=df['weight'].fillna(df['weight'].mean())
df['sight_left']=df['sight_left'].fillna(df['sight_left'].mean())
df['SBP']=df['SBP'].fillna(df['SBP'].mean())
df['DBP']=df['DBP'].fillna(df['DBP'].mean())
df['BLDS']=df['BLDS'].fillna(df['BLDS'].mean())
df['tot_chole']=df['tot_chole'].fillna(df['tot_chole'].mean())
df['triglyceride']=df['triglyceride'].fillna(df['triglyceride'].mean())
df['SGOT_AST']=df['SGOT_AST'].fillna(df['SGOT_AST'].mean())

```

```
df.isna().sum()
```

```
sex                0
age                0
height             0
weight             0
waistline          0
sight_left         0
sight_right        0
hear_left          0
hear_right         0
SBP                0
DBP                0
BLDS               0
tot_chole          0
HDL_chole          0
LDL_chole          0
triglyceride       0
hemoglobin         0
urine_protein      0
serum_creatinine   0
SGOT_AST           0
SGOT_ALT           0
gamma_GTP          0
DRK_YN             0
dtype: int64
```

```
df.dtypes
```

```
sex                  object
age                   int64
height              float64
weight              float64
waistline           float64
sight_left          float64
sight_right         float64
hear_left             int64
hear_right            int64
SBP                 float64
DBP                 float64
BLDS                float64
tot_chole           float64
HDL_chole             int64
LDL_chole             int64
triglyceride        float64
hemoglobin          float64
urine_protein         int64
serum_creatinine    float64
SGOT_AST            float64
SGOT_ALT              int64
gamma_GTP             int64
DRK_YN               object
dtype: object
```

Encoding

```
1 from sklearn.preprocessing import LabelEncoder
2 le=LabelEncoder()
3 df['sex']=le.fit_transform(df['sex'])
4 df['DRK_YN']=le.fit_transform(df['DRK_YN'])
```

```
1 df.columns
```

```
Index(['sex', 'age', 'height', 'weight', 'waistline', 'sight_left',
       'sight_right', 'hear_left', 'hear_right', 'SBP', 'DBP', 'BLDS',
       'tot_chole', 'HDL_chole', 'LDL_chole', 'triglyceride', 'hemoglobin',
       'urine_protein', 'serum_creatinine', 'SGOT_AST', 'SGOT_ALT',
       'gamma_GTP', 'DRK_YN'],
      dtype='object')
```

```
1 df.dtypes
```

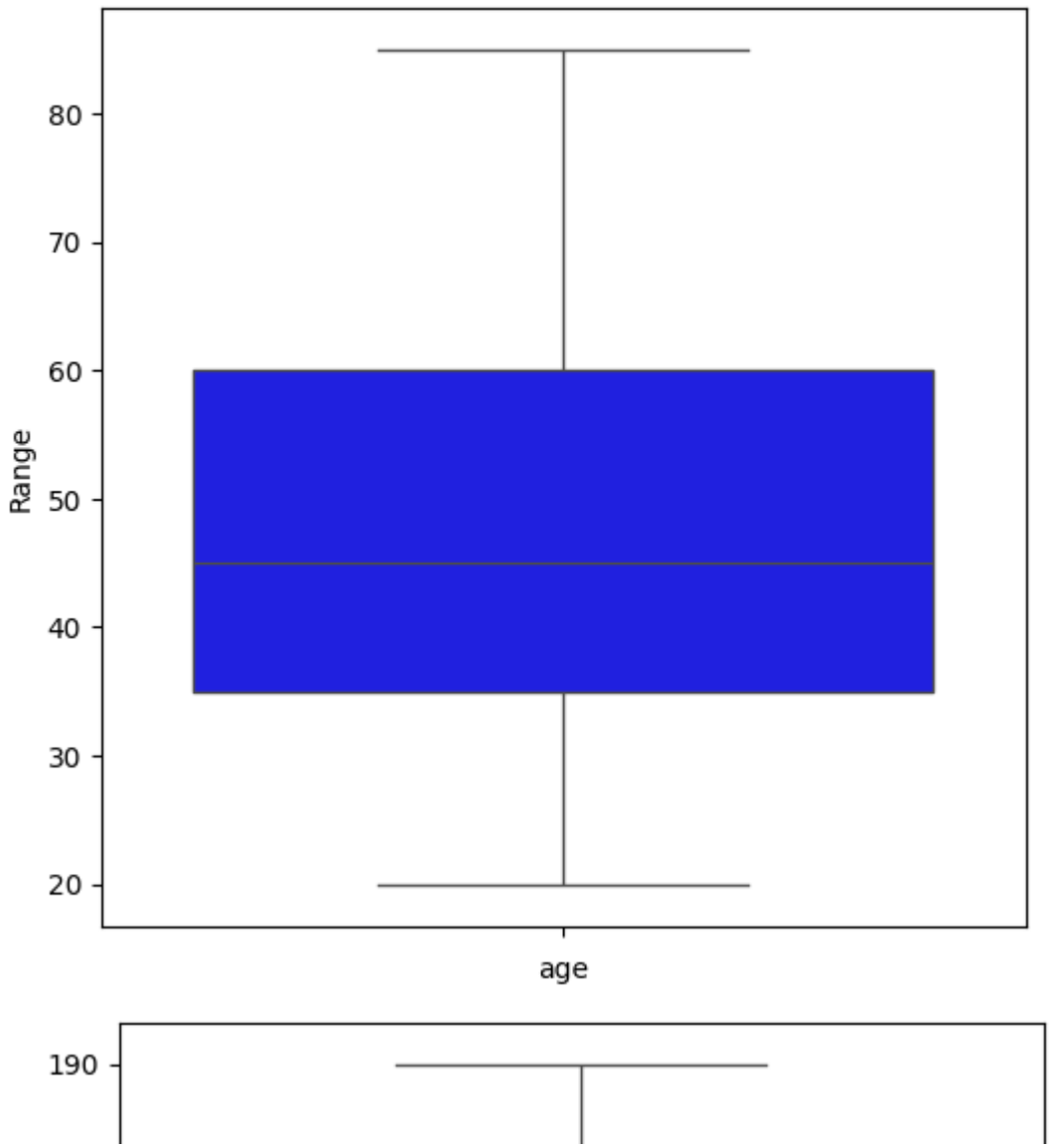
```
sex                  int64
age                  int64
height             float64
weight             float64
waistline          float64
sight_left         float64
sight_right        float64
hear_left            int64
hear_right           int64
SBP                float64
DBP                float64
BLDS               float64
tot_chole          float64
```

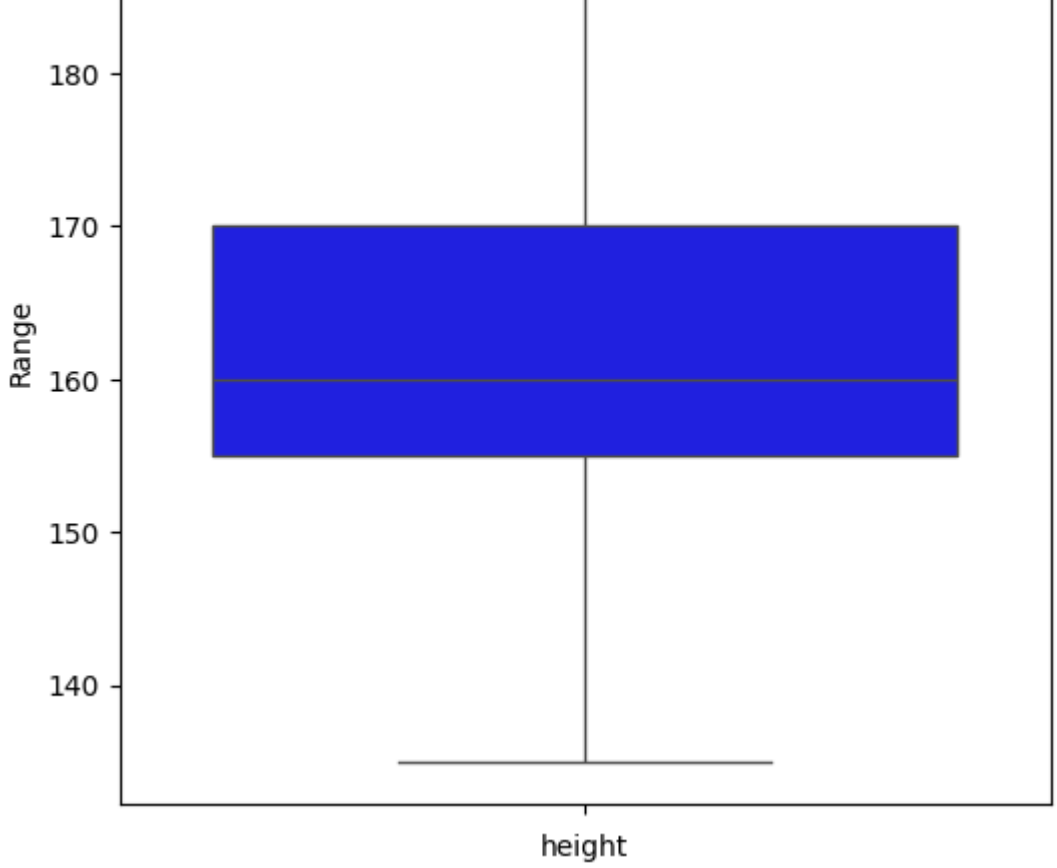
```
tot_chole           float64
HDL_chole             int64
LDL_chole             int64
triglyceride        float64
hemoglobin          float64
urine_protein         int64
serum_creatinine    float64
SGOT_AST            float64
SGOT_ALT              int64
gamma_GTP             int64
DRK_YN                int64
dtype: object
```

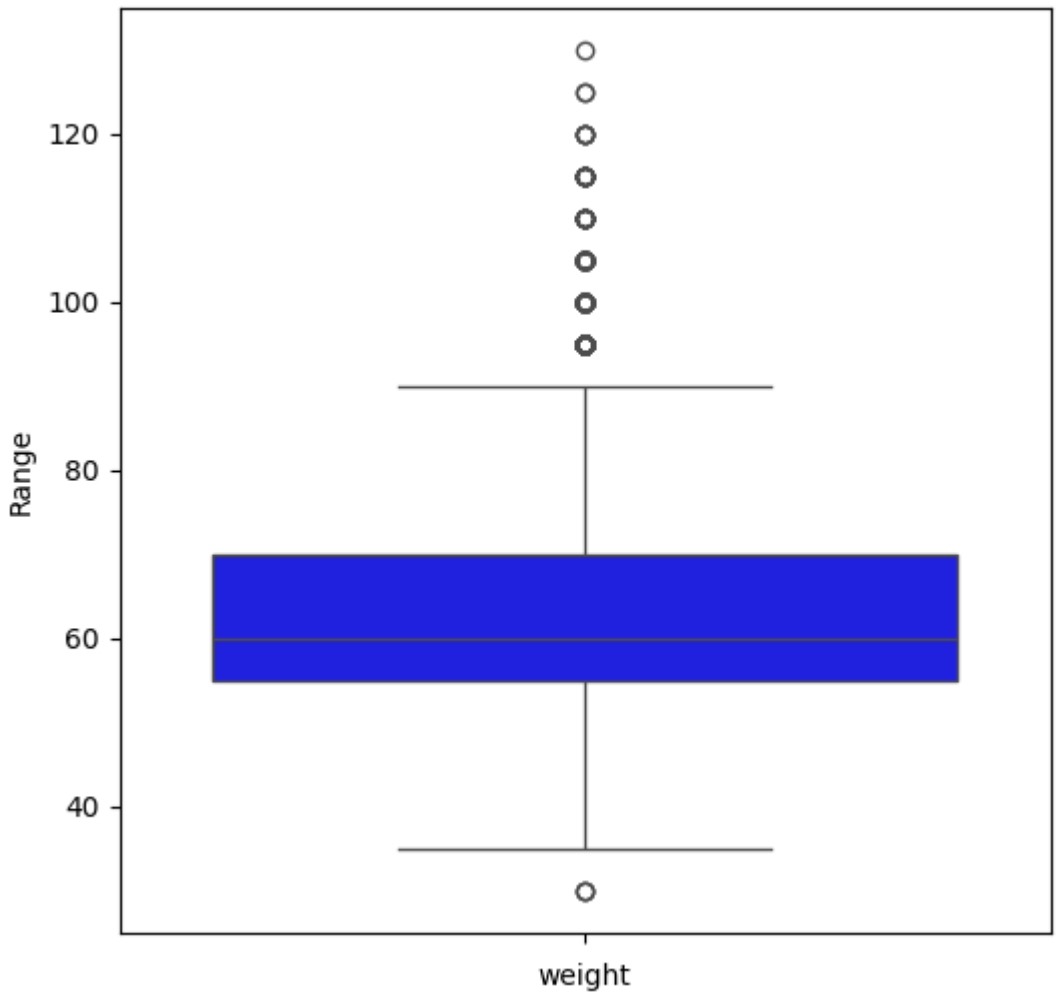
Finding & Correcting Outlayers

```
for i in['age','height','weight','waistline','sight_left','sight_right','hear_left','hear
  plt.figure(figsize=(6,6))
  sns.boxplot(df[i],color='blue')
  plt.xlabel(i)
  plt.ylabel('Range')
```

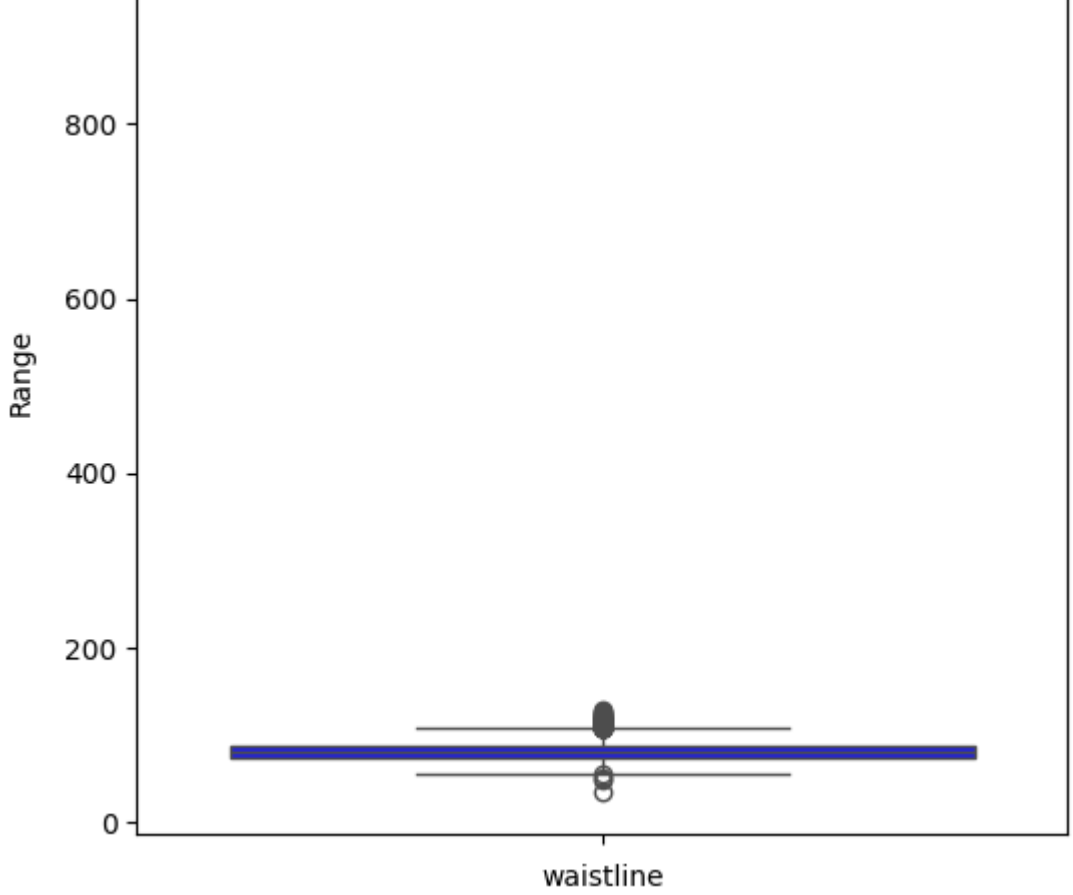
```
<ipython-input-18-0e1e8f45fcc1>:2: RuntimeWarning: More than 20 figures have been openec
  plt.figure(figsize=(6,6))
```

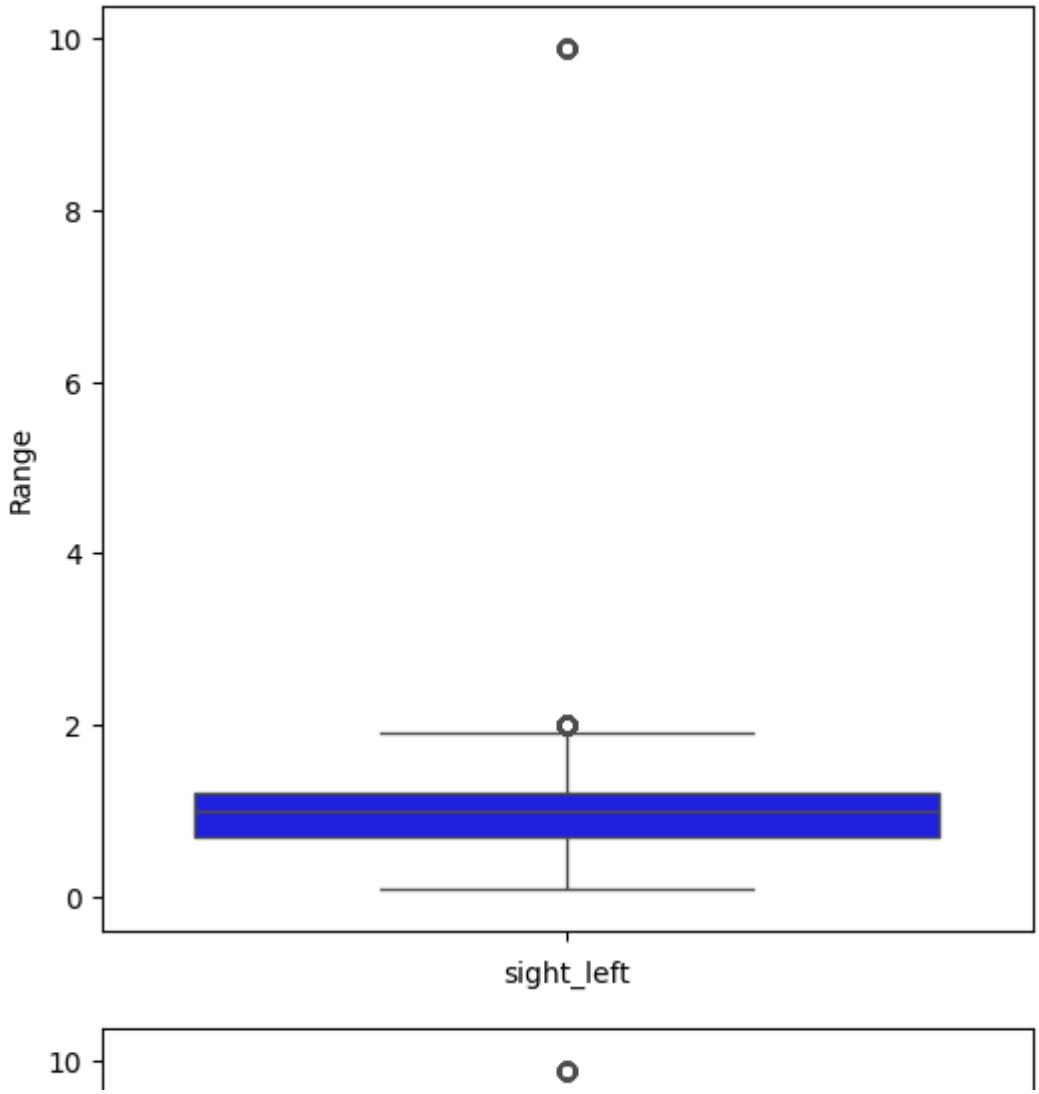



180
170
160
150
140
Range
height

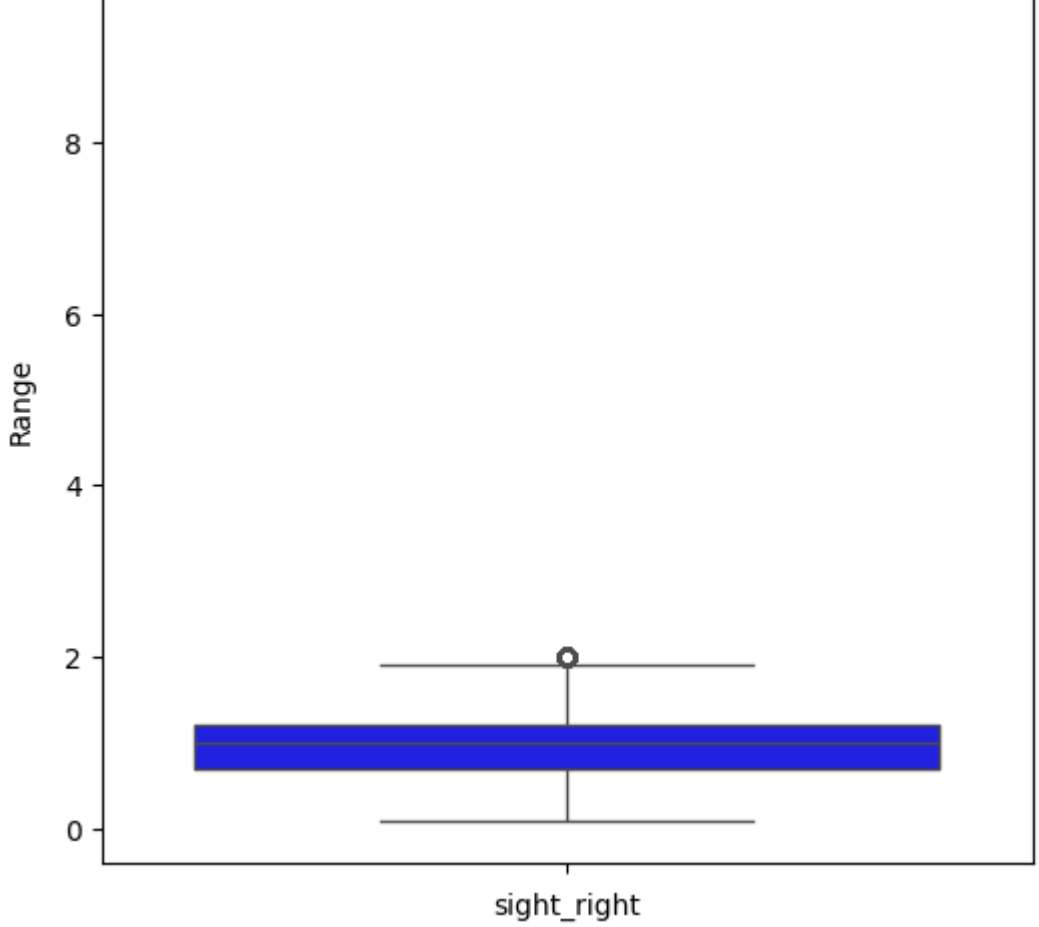

120
100
80
60
40
Range
weight

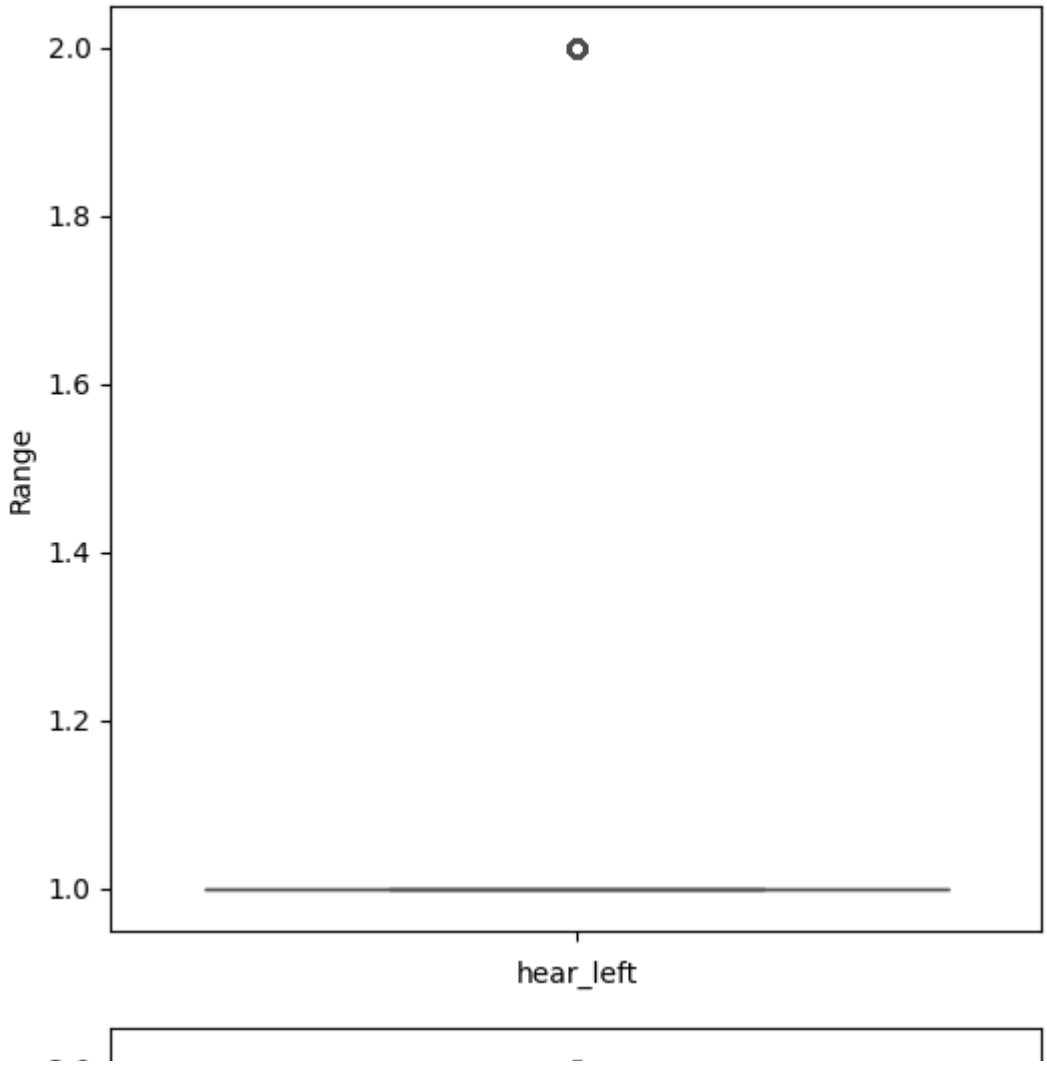

1000

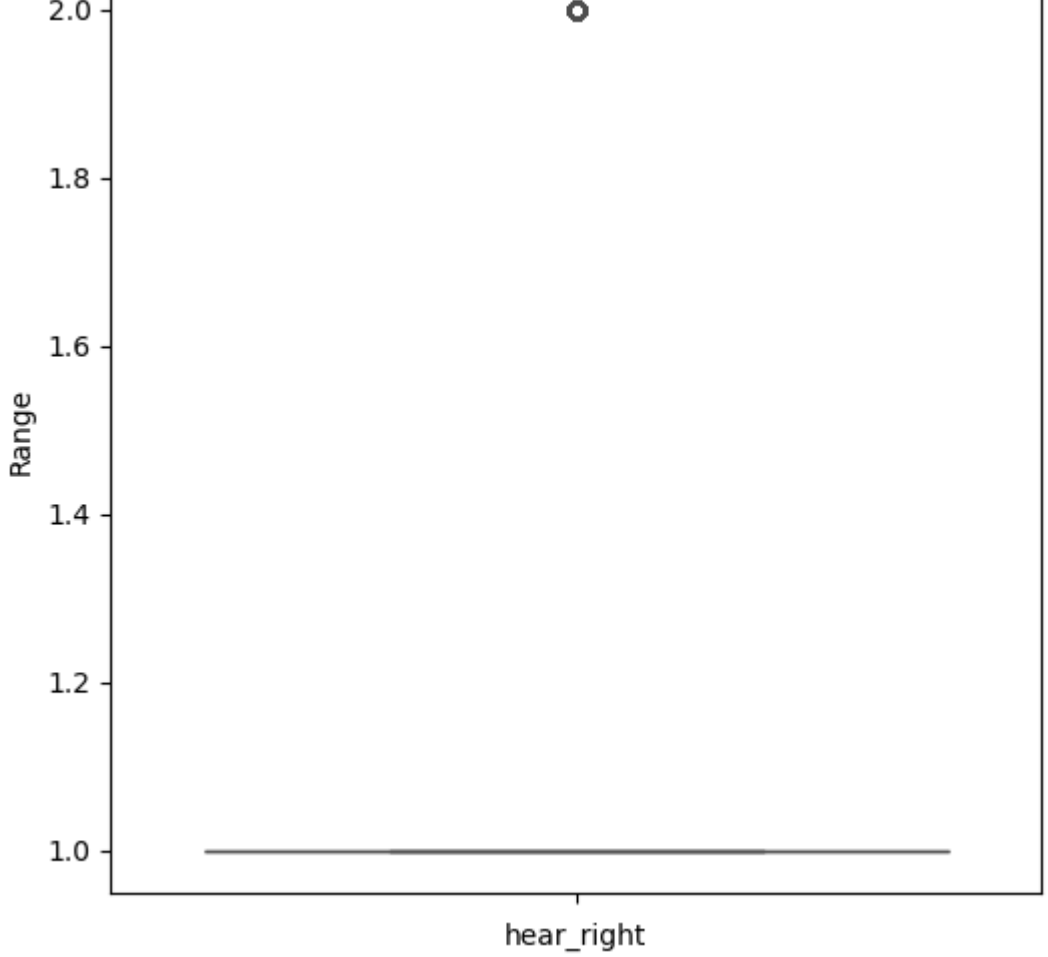

800
600
Range
400
200
0
waistline

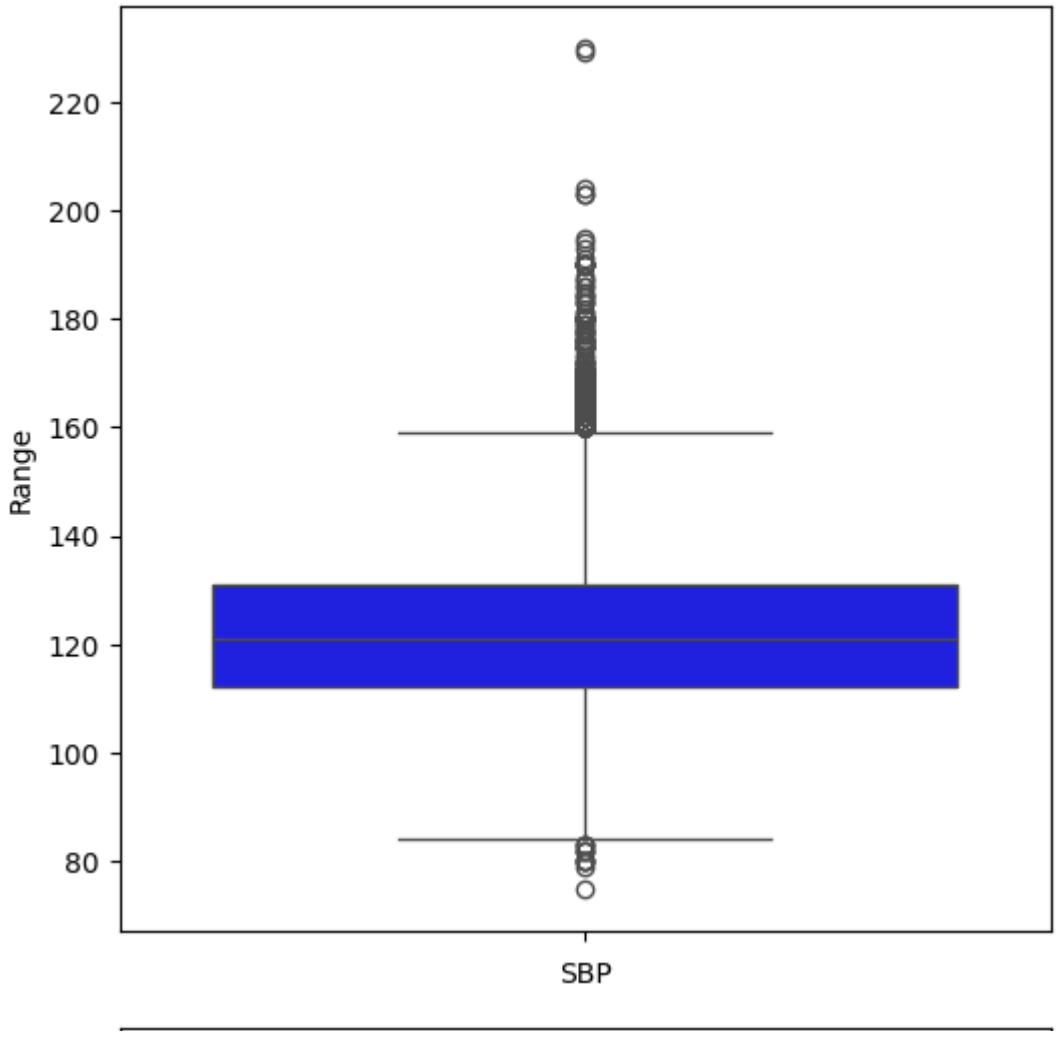

10
8
6
Range
4
2
0
sight_left

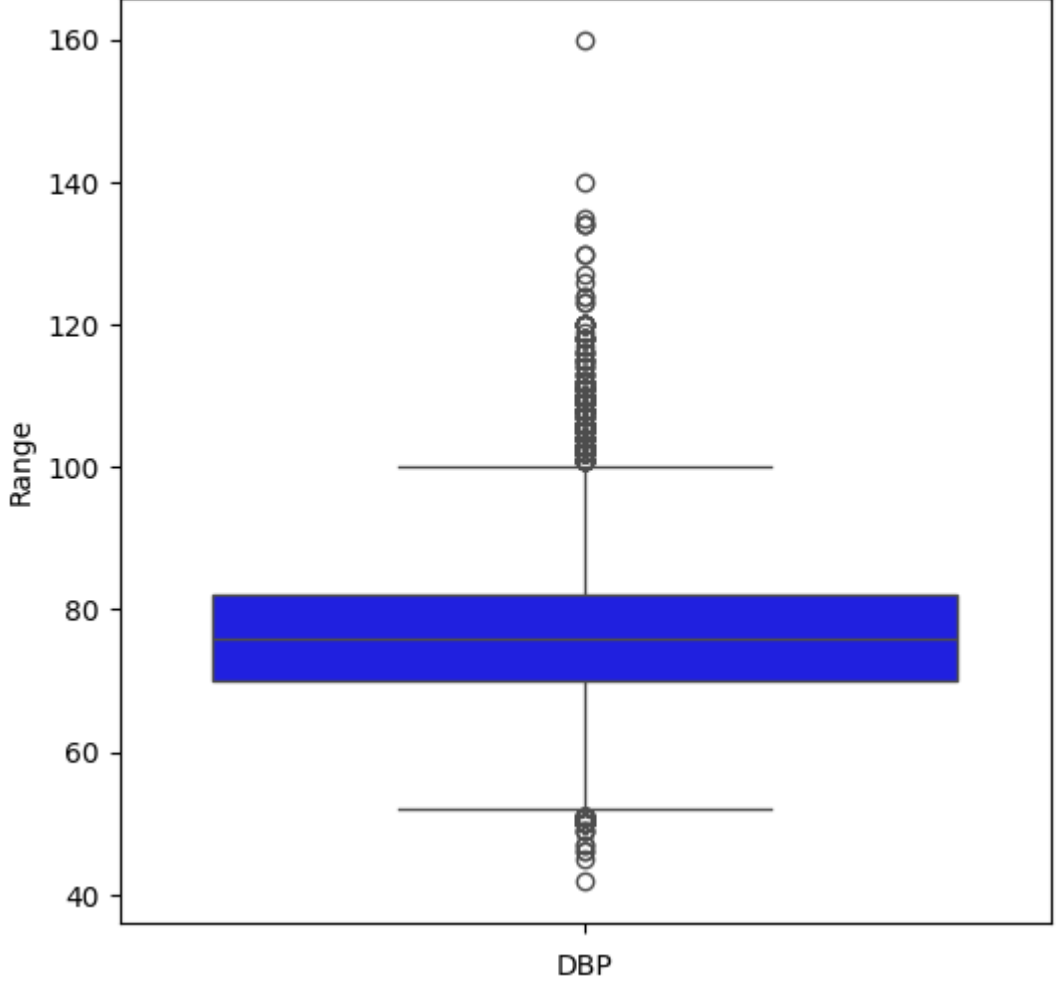
10

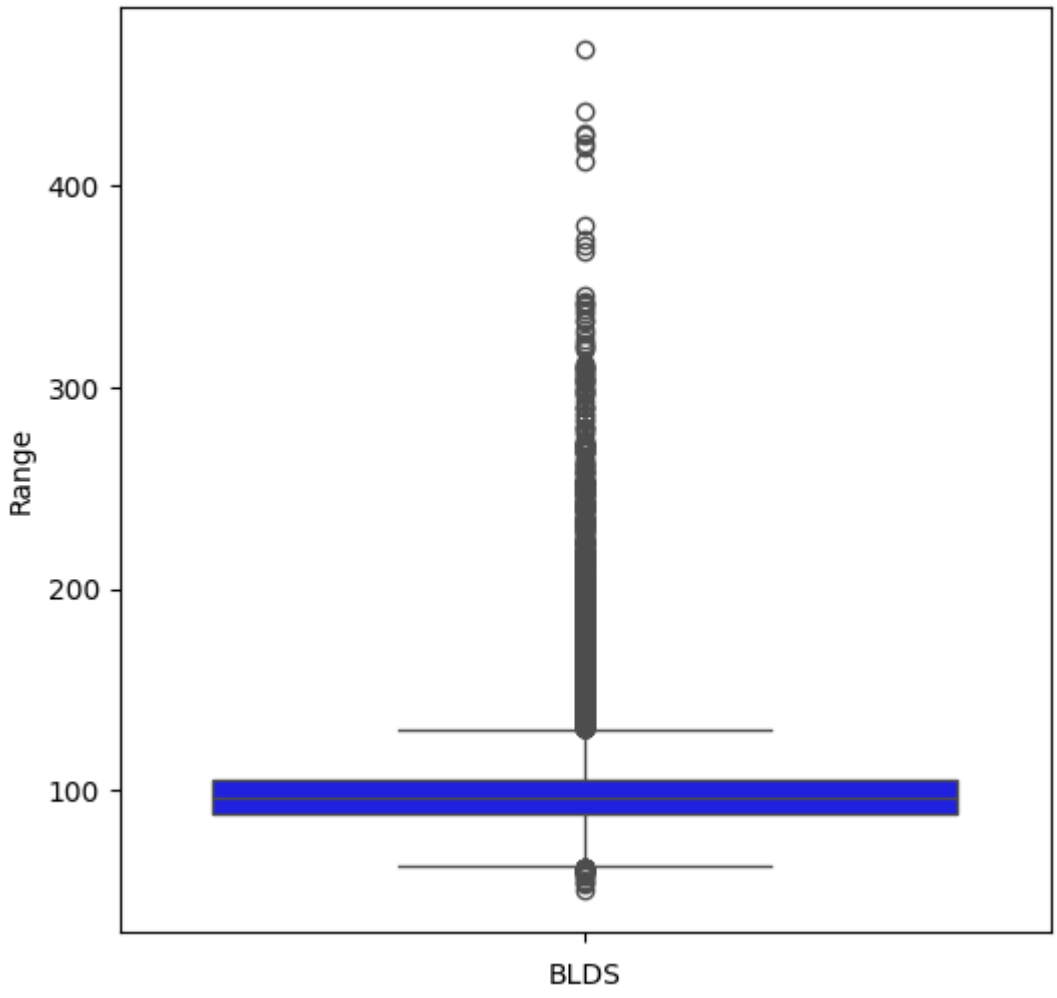

8
6
Range
4
2
0
sight_right

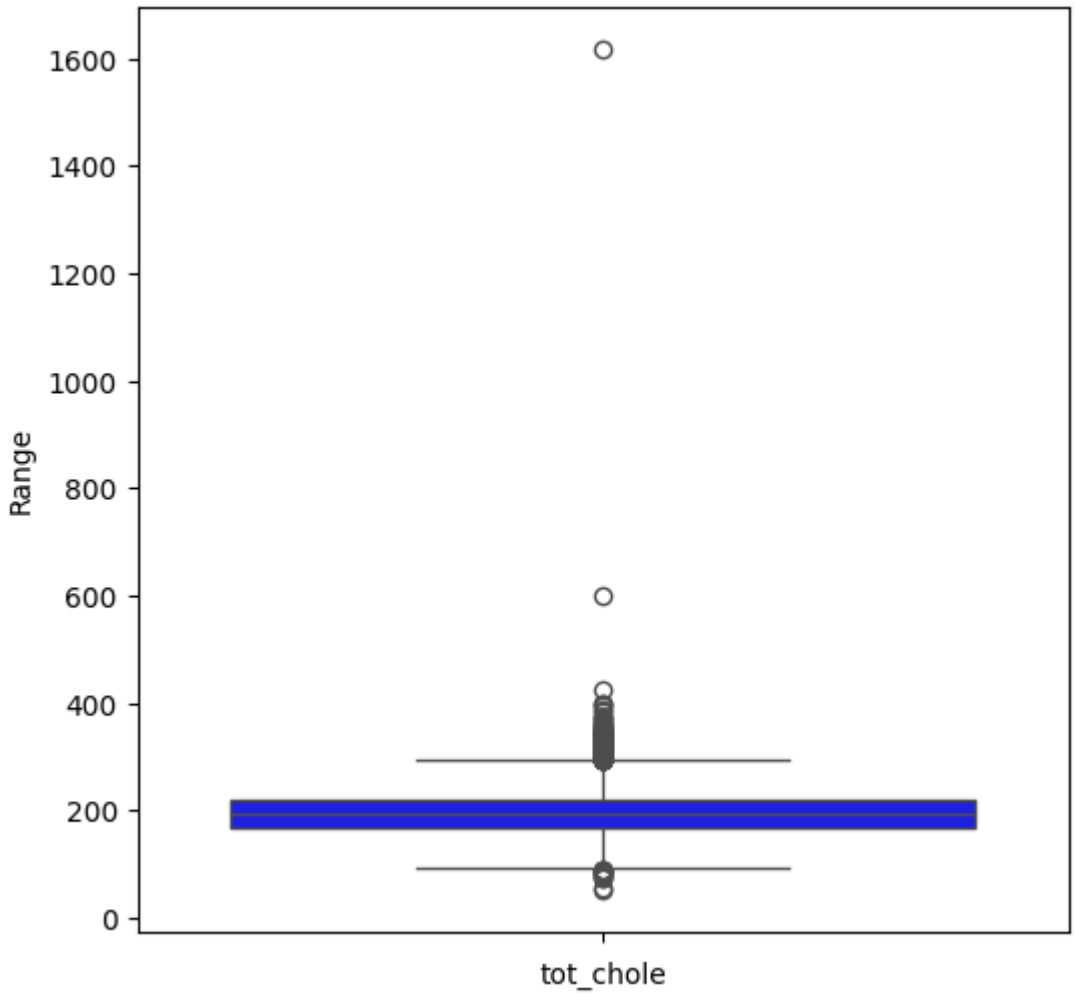

2.0
1.8
1.6
Range
1.4
1.2
1.0
hear_left

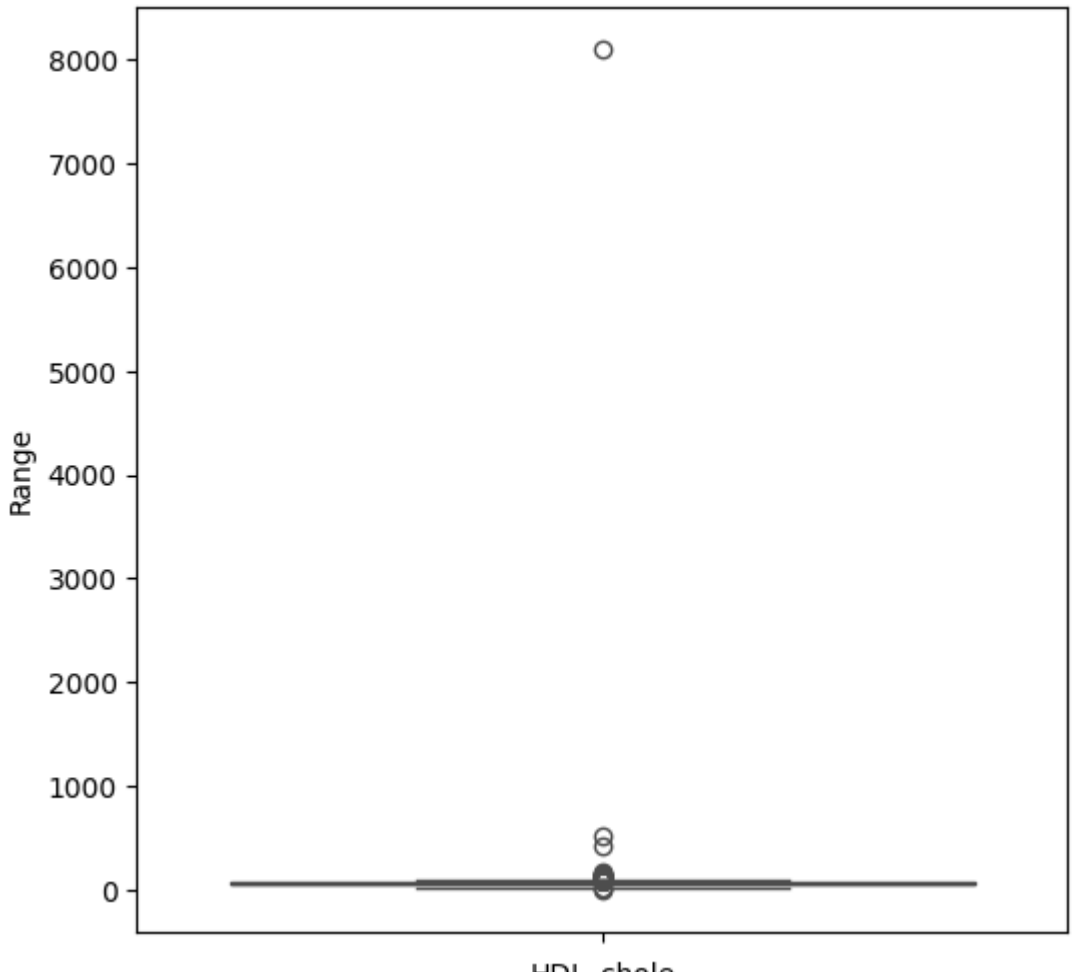

2.0
1.8
1.6
1.4
1.2
1.0
Range
hear_right

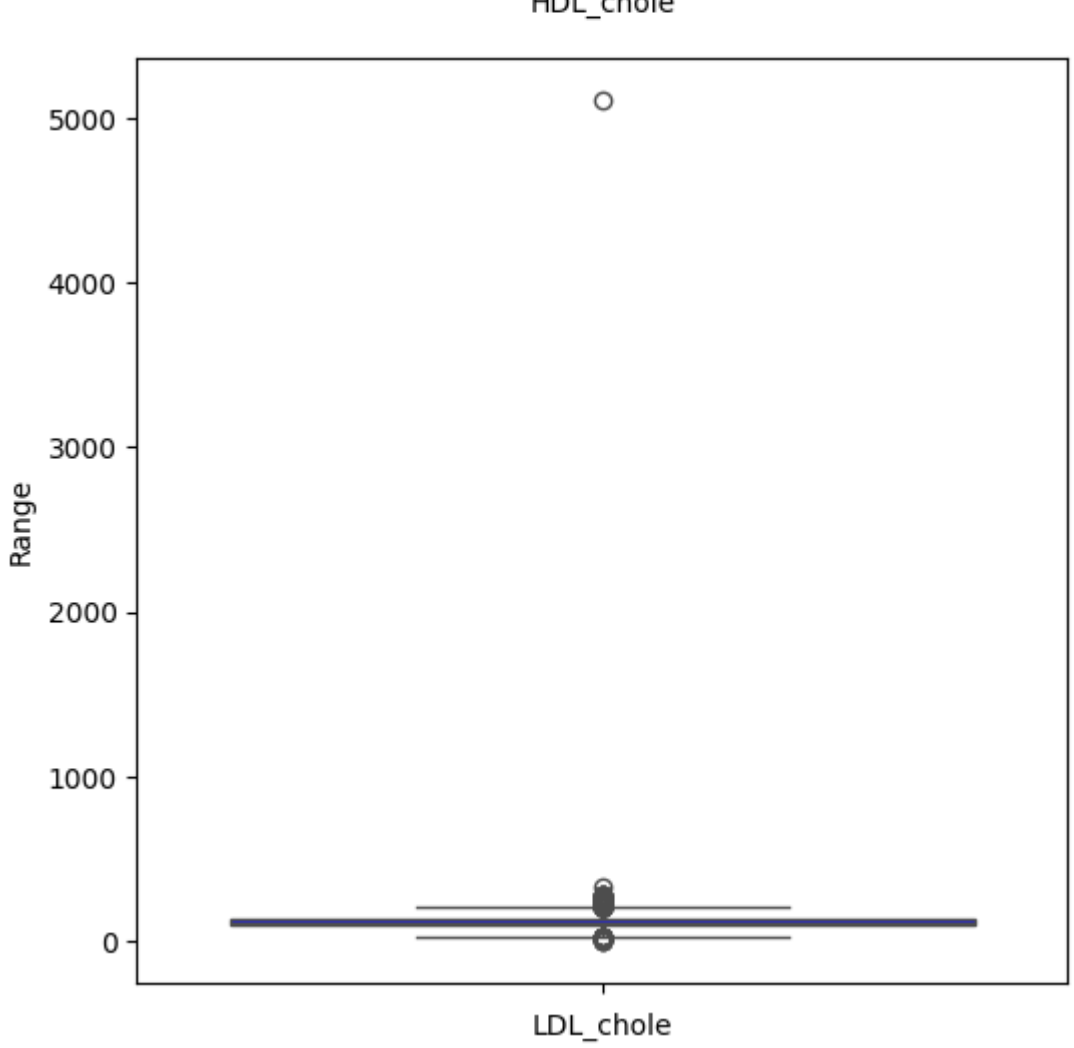

220
200
180
160
140
120
100
80
Range
SBP

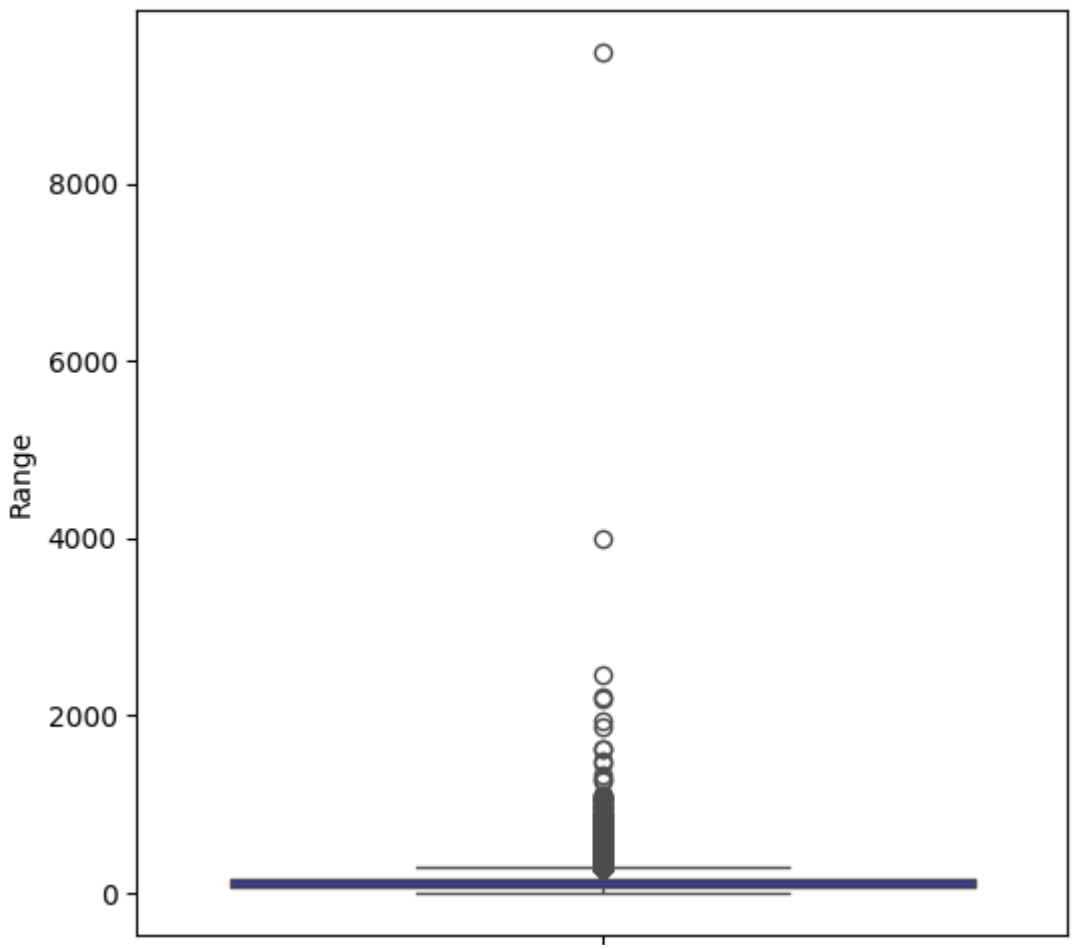

160
140
120
100
80
60
40
Range
DBP

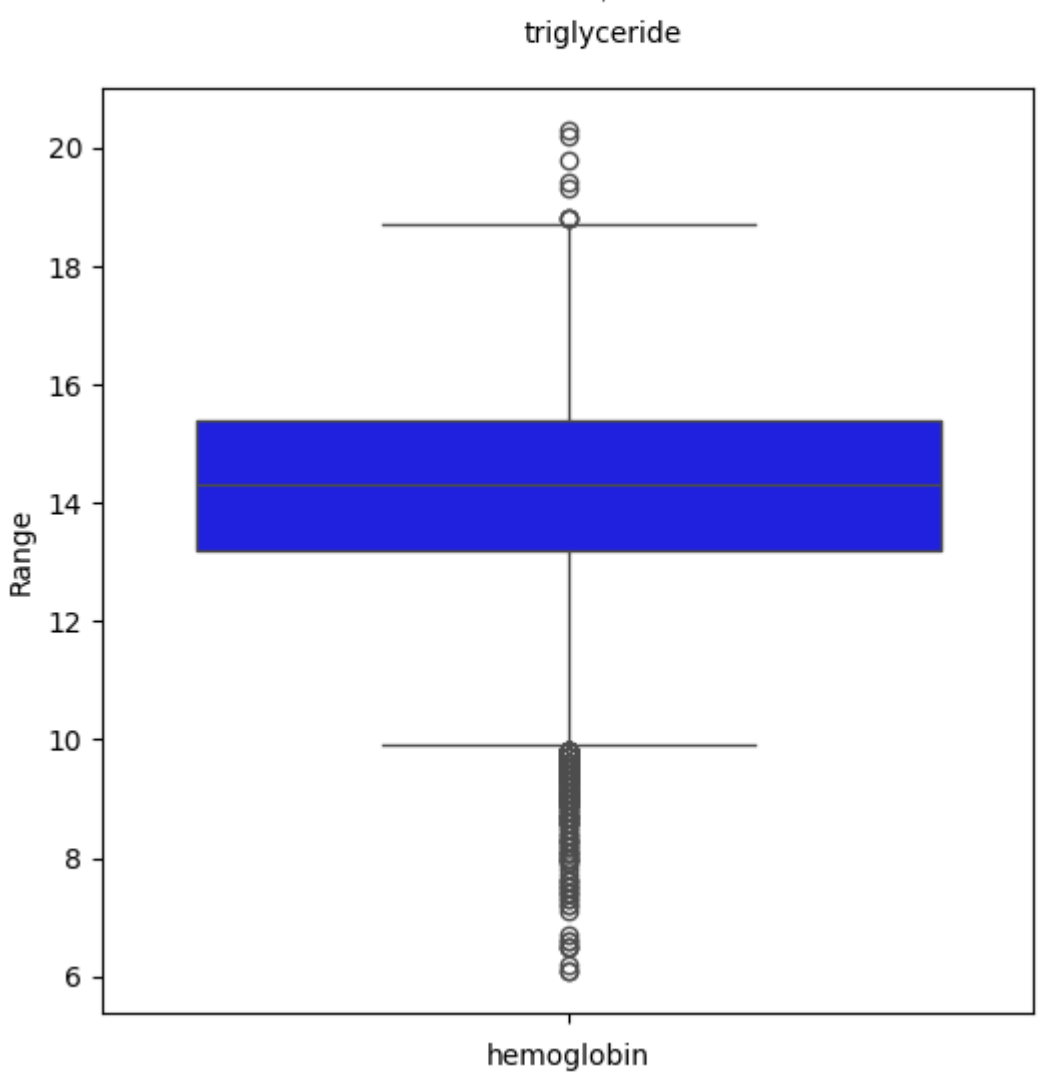

400
300
200
100
Range
BLDS

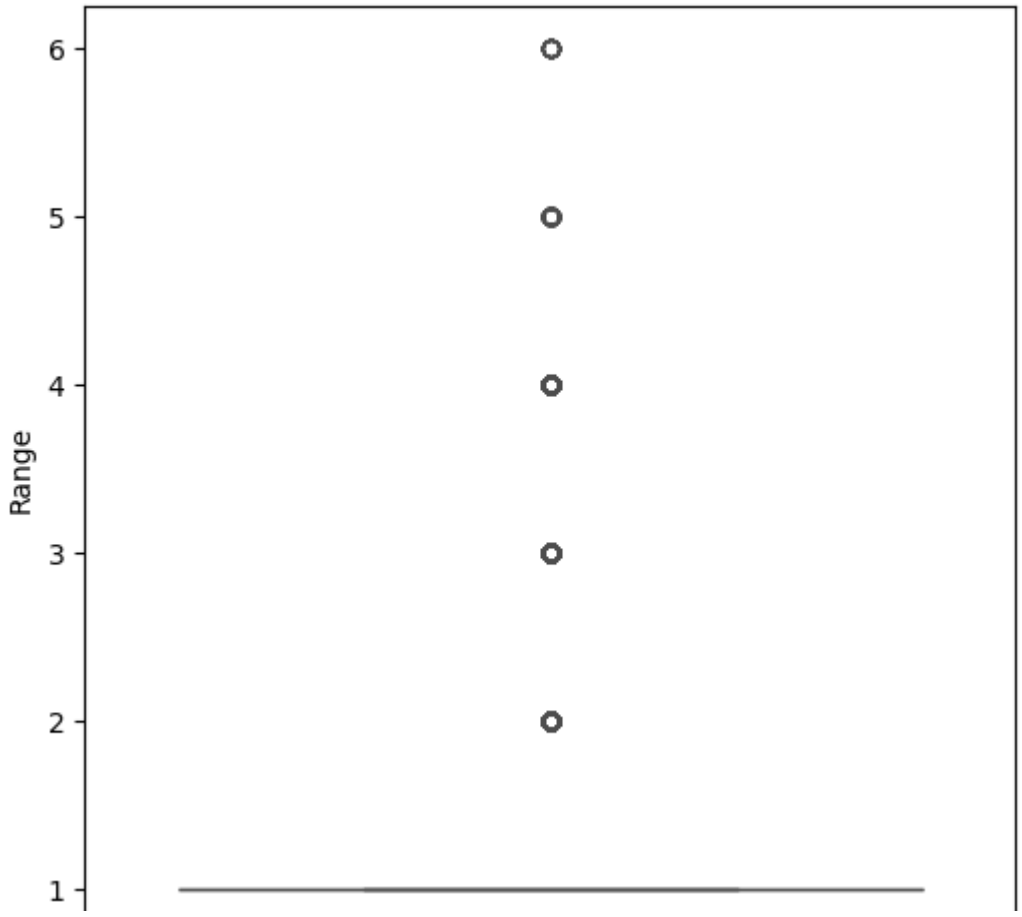

1600
1400
1200
1000
800
600
400
200
0
Range
tot_chole

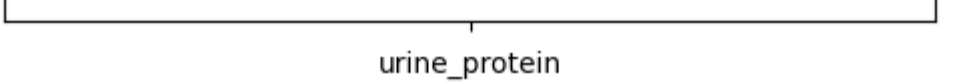

8000
7000
6000
5000
4000
3000
2000
1000
0
Range


HDL_chole
5000
4000
3000
2000
1000
0
Range
LDL_chole

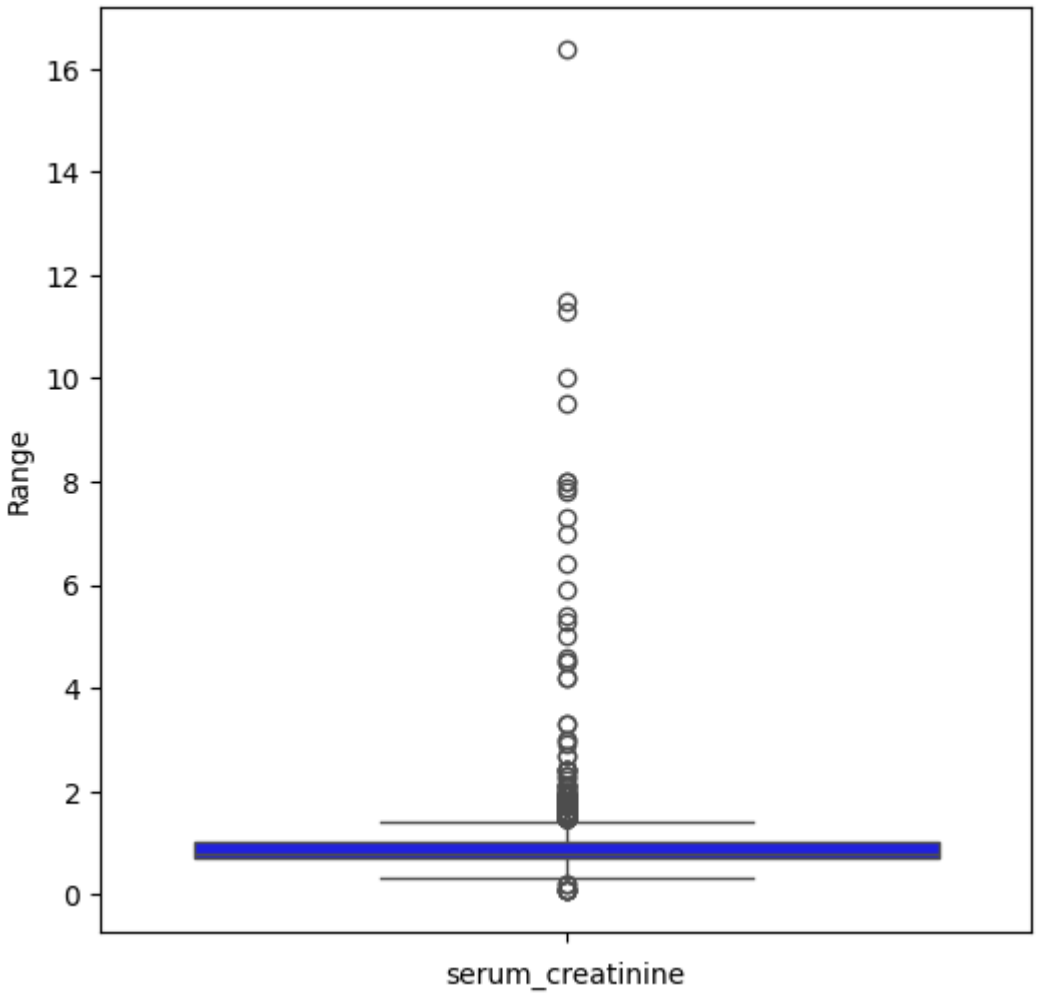

8000
6000
4000
2000
0
Range


triglyceride
Range
20
18
16
14
12
10
8
6
hemoglobin

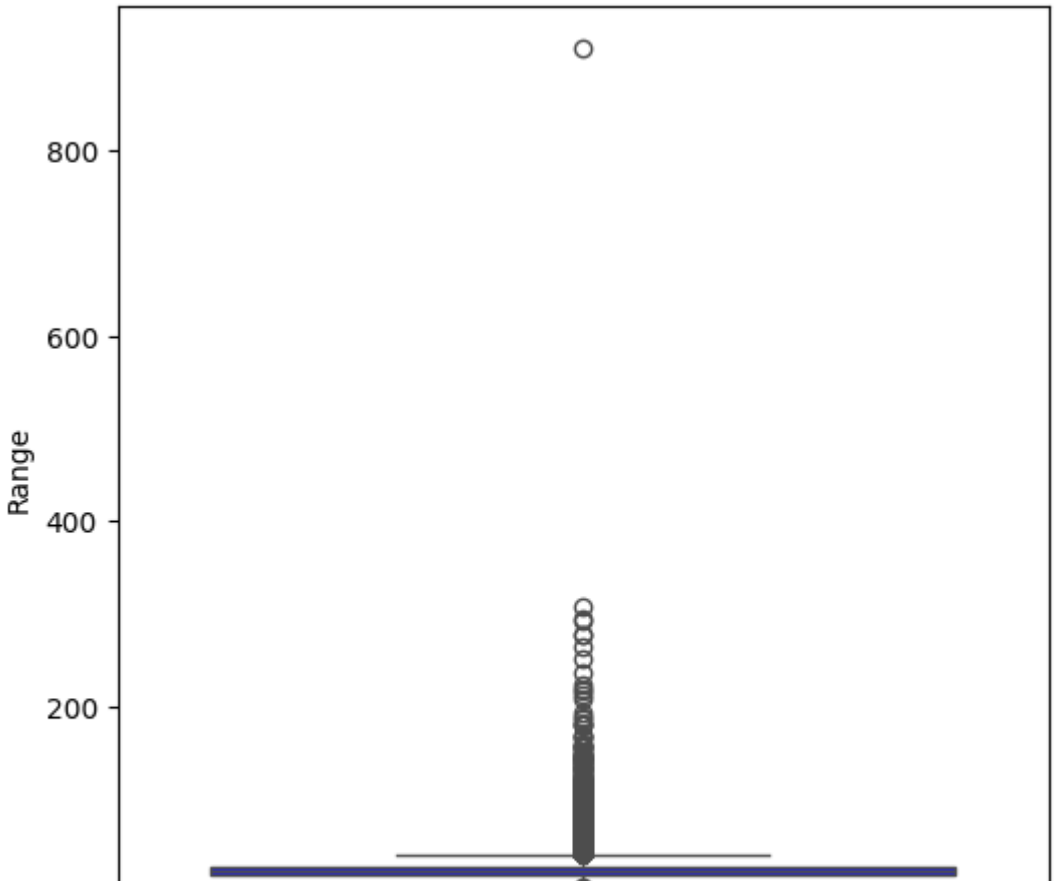

Range
6
5
4
3
2
1

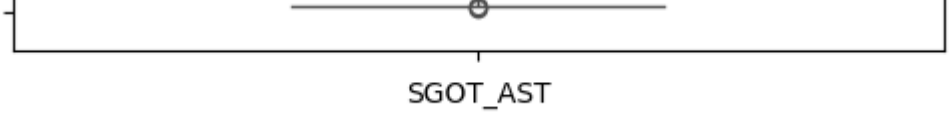


urine_protein

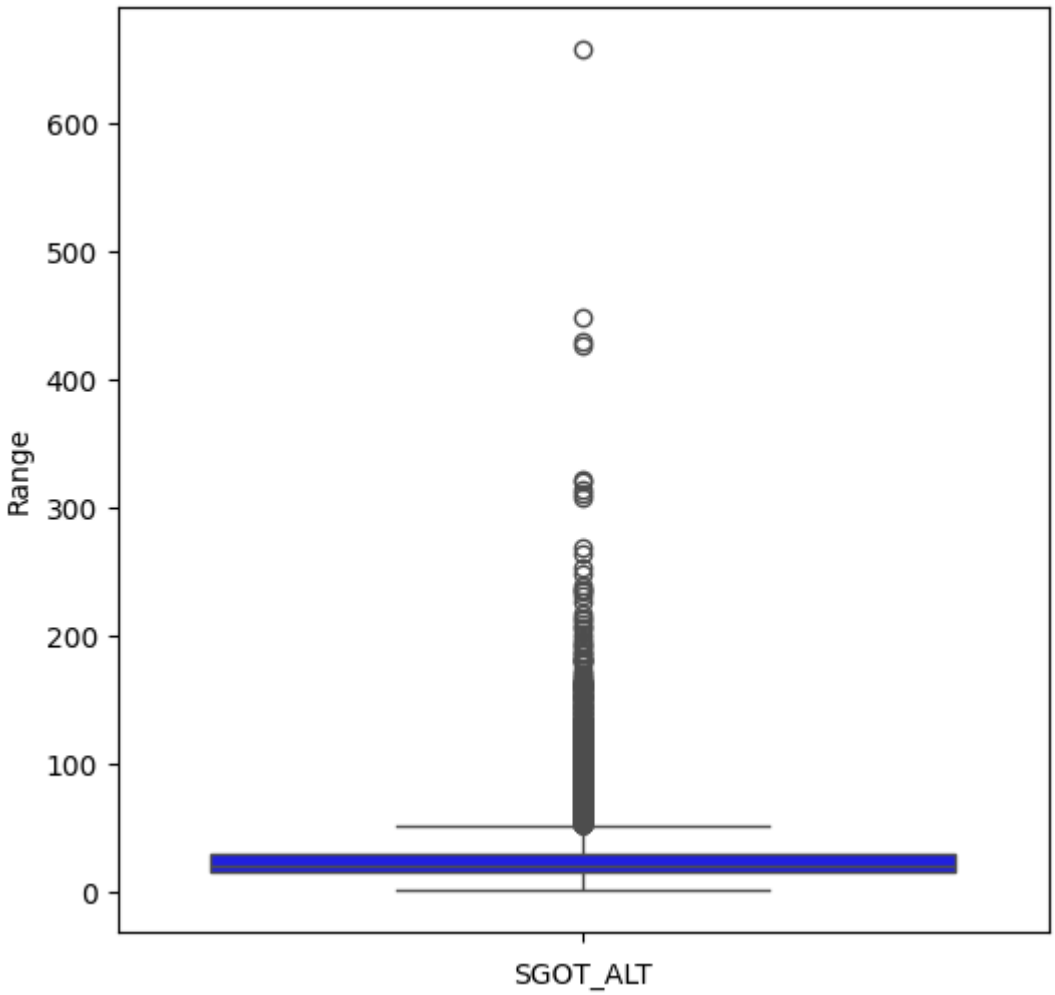


serum_creatinine

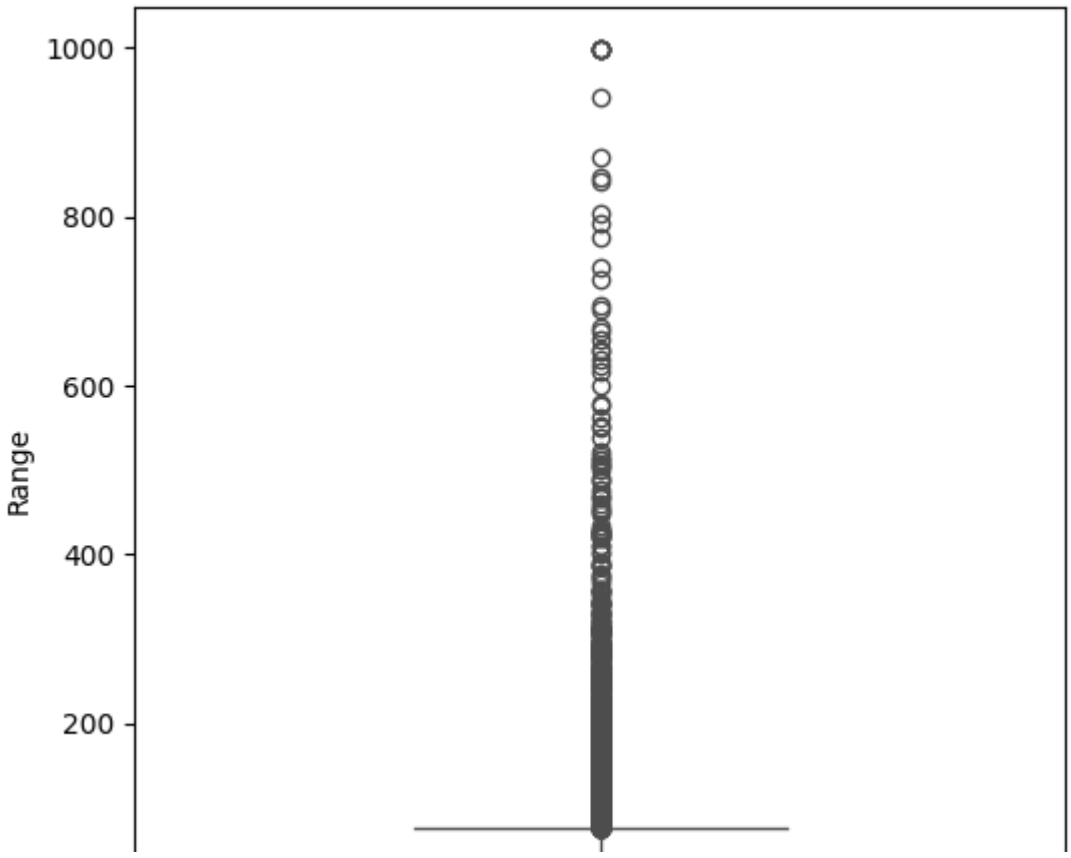


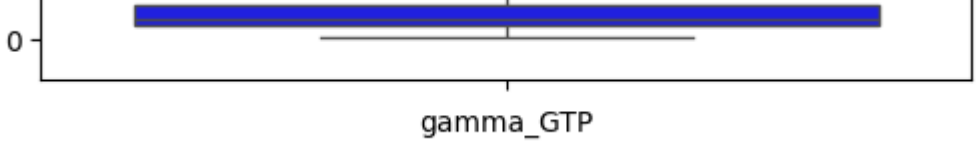


SGOT_AST



SGOT_ALT






```
df = df.select_dtypes(include=['number'])
q1=df.quantile(0.25)
q2=df.quantile(0.75)
IQR=q2-q1                  #Interquartile Range
max_limit=q2+(1.5*IQR)
min_limit=q1-(1.5*IQR)
```

```
df=pd.DataFrame(np.where(df>max_limit,max_limit,(np.where(df<min_limit,min_limit,df))),cc
```

```
df.shape
```

```
(23999, 23)
```

```
df.nunique()
```

```
sex                  2
age                 14
height              13
weight              15
waistline          460
sight_left          19
sight_right         19
hear_left            1
hear_right           1
SBP                 79
DBP                 50
BLDS                71
tot_chole          201
HDL_chole           80
LDL_chole          185
triglyceride       275
hemoglobin          91
urine_protein        1
serum_creatinine    14
SGOT_AST            39
SGOT_ALT            52
gamma_GTP           74
DRK_YN               2
dtype: int64
```

```
# for i in['age','height','weight','waistline','sight_left','sight_right','SBP','DBP','BL
#   plt.figure(figsize=(6,6))
#   sns.boxplot(df[i],color='blue')
#   plt.xlabel(i)
#   plt.ylabel('Range')
```

```
df['hear_left']
```

```
0        1.0
1        1.0
2        1.0
3        1.0
4        1.0
        ...
23994    1.0
23995    1.0
23996    1.0
23997    1.0
23998    1.0
Name: hear_left, Length: 23999, dtype: float64
```

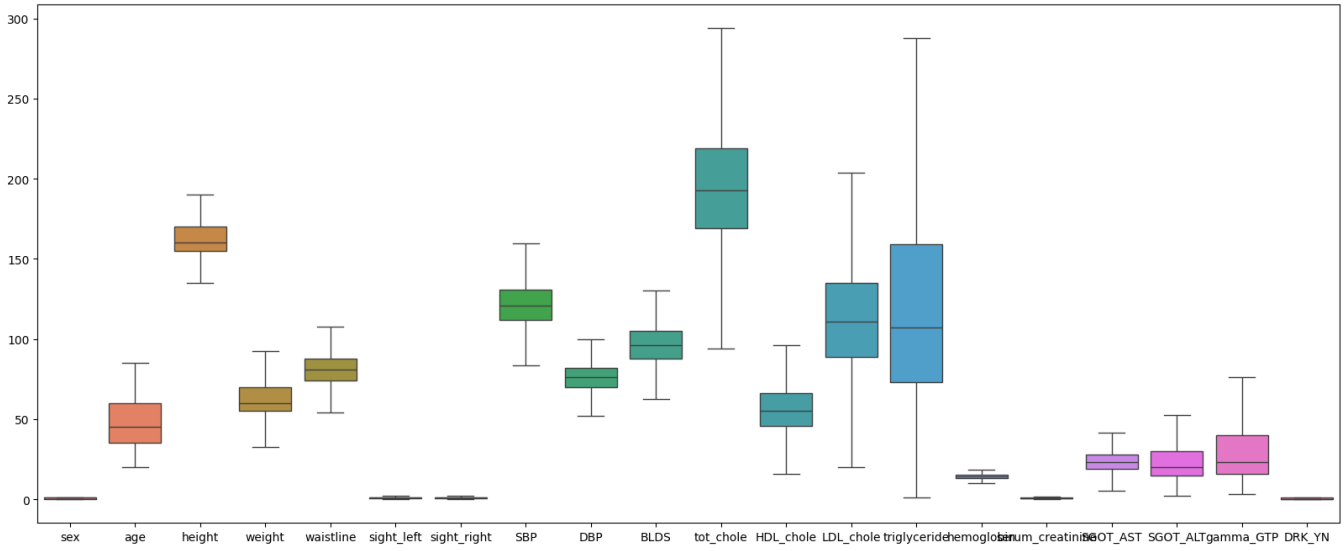
```
1 df['hear_left'].value_counts()
```

```
hear_left
1.0    23999
Name: count, dtype: int64
```

```
1 df.drop(['hear_left','hear_right','urine_protein'],axis=1,inplace=True)
```

```
1 plt.figure(figsize=(20,8))
2 sns.boxplot(data=df)
```

```
<Axes: >
```



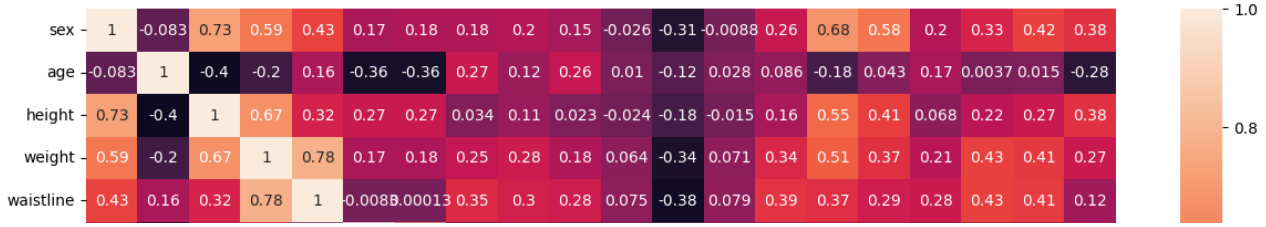
Checking Co-relation

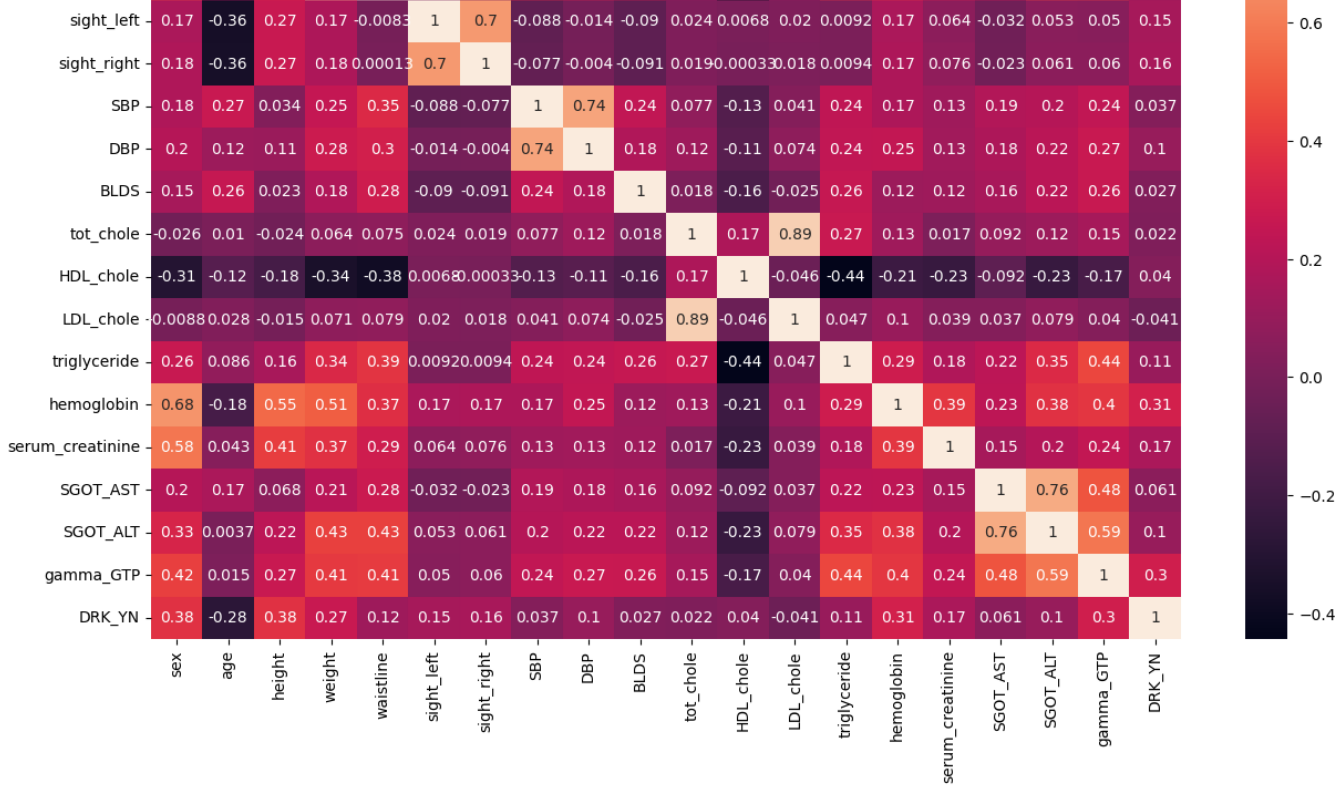
```
1 df.corr()
```

	sex	age	height	weight	waistline	sight_left	sight_r
sex	1.000000	-0.083209	0.726752	0.590532	0.432042	0.167185	0.17
age	-0.083209	1.000000	-0.395952	-0.197100	0.164387	-0.360460	-0.35
height	0.726752	-0.395952	1.000000	0.668320	0.322849	0.268216	0.27
weight	0.590532	-0.197100	0.668320	1.000000	0.783528	0.171450	0.17
waistline	0.432042	0.164387	0.322849	0.783528	1.000000	-0.008308	0.00
sight_left	0.167185	-0.360460	0.268216	0.171450	-0.008308	1.000000	0.69
sight_right	0.178484	-0.356387	0.270467	0.176304	0.000132	0.698550	1.00
SBP	0.180282	0.271473	0.033525	0.252190	0.347212	-0.087821	-0.07
DBP	0.203149	0.115360	0.111095	0.280387	0.304965	-0.014214	-0.00
BLDS	0.151638	0.259969	0.022815	0.184589	0.284012	-0.089856	-0.09
tot_chole	-0.025790	0.010458	-0.023743	0.064095	0.074761	0.023682	0.01
HDL_chole	-0.307030	-0.118202	-0.182061	-0.343599	-0.376161	0.006797	-0.00
LDL_chole	-0.008804	0.027586	-0.014537	0.071464	0.078892	0.019791	0.01
triglyceride	0.262788	0.086186	0.155686	0.344221	0.387878	0.009169	0.00
hemoglobin	0.680379	-0.177468	0.547527	0.514122	0.367610	0.168812	0.17
serum_creatinine	0.584680	0.042963	0.413638	0.374681	0.292700	0.064146	0.07
SGOT_AST	0.202628	0.170429	0.068020	0.211990	0.276781	-0.031622	-0.02
SGOT_ALT	0.326451	0.003721	0.224380	0.428773	0.429396	0.052933	0.06
gamma_GTP	0.419859	0.014914	0.272427	0.406772	0.408744	0.049675	0.06
DRK_YN	0.377456	-0.278262	0.375580	0.268724	0.124624	0.151138	0.15

Heatmap

```
1 plt.figure(figsize=(15,10))
2 sns.heatmap(df.corr(),annot=True)
3 plt.show()
```






```
1 df_corr = df.corr()
2 df_corr
```

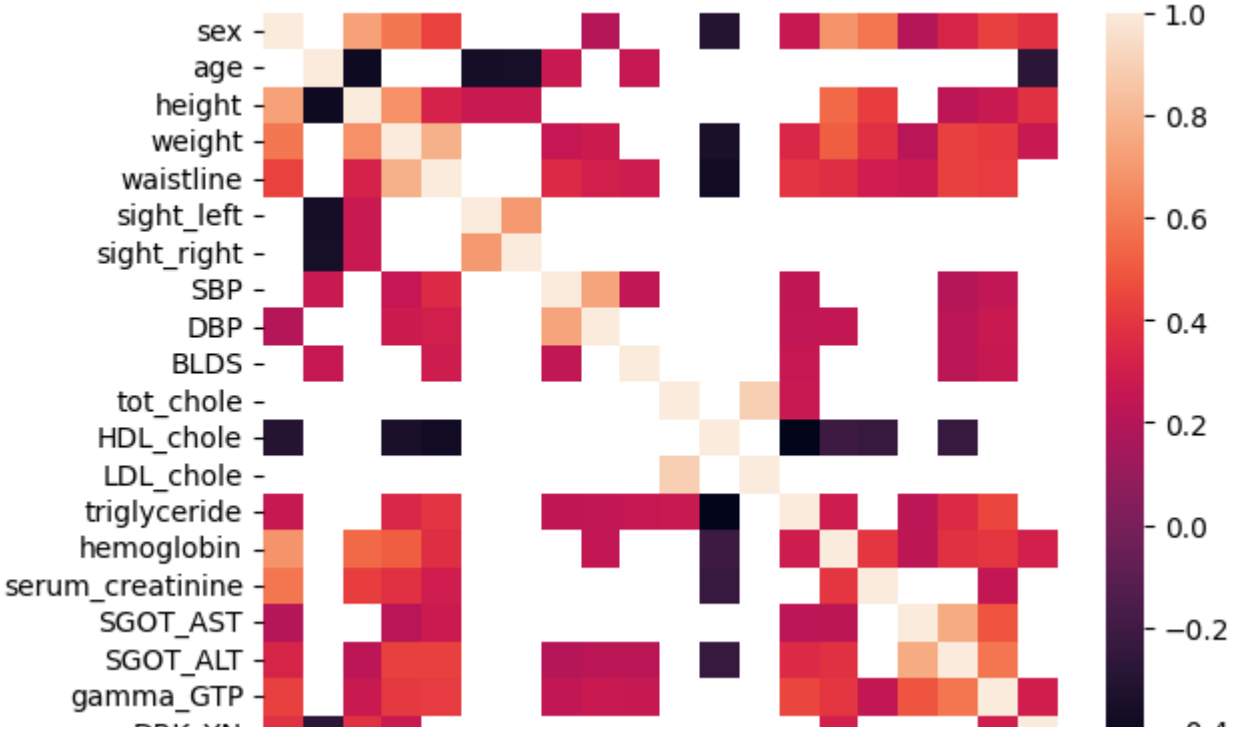
	sex	age	height	weight	waistline	sight_left	sight_r
sex	1.000000	-0.083209	0.726752	0.590532	0.432042	0.167185	0.17
age	-0.083209	1.000000	-0.395952	-0.197100	0.164387	-0.360460	-0.35
height	0.726752	-0.395952	1.000000	0.668320	0.322849	0.268216	0.27
weight	0.590532	-0.197100	0.668320	1.000000	0.783528	0.171450	0.17

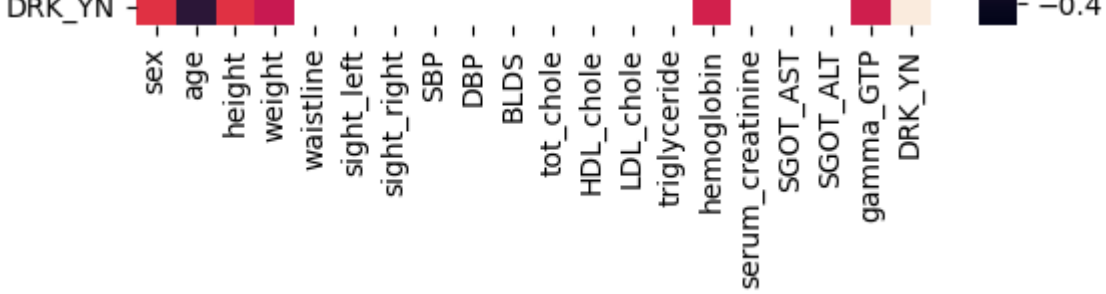
waistline	0.432042	0.164387	0.322849	0.783528	1.000000	-0.008308	0.00
sight_left	0.167185	-0.360460	0.268216	0.171450	-0.008308	1.000000	0.69
sight_right	0.178484	-0.356387	0.270467	0.176304	0.000132	0.698550	1.00
SBP	0.180282	0.271473	0.033525	0.252190	0.347212	-0.087821	-0.07
DBP	0.203149	0.115360	0.111095	0.280387	0.304965	-0.014214	-0.00
BLDS	0.151638	0.259969	0.022815	0.184589	0.284012	-0.089856	-0.09
tot_chole	-0.025790	0.010458	-0.023743	0.064095	0.074761	0.023682	0.01
HDL_chole	-0.307030	-0.118202	-0.182061	-0.343599	-0.376161	0.006797	-0.00
LDL_chole	-0.008804	0.027586	-0.014537	0.071464	0.078892	0.019791	0.01
triglyceride	0.262788	0.086186	0.155686	0.344221	0.387878	0.009169	0.00
hemoglobin	0.680379	-0.177468	0.547527	0.514122	0.367610	0.168812	0.17
serum_creatinine	0.584680	0.042963	0.413638	0.374681	0.292700	0.064146	0.07
SGOT_AST	0.202628	0.170429	0.068020	0.211990	0.276781	-0.031622	-0.02
SGOT_ALT	0.326451	0.003721	0.224380	0.428773	0.429396	0.052933	0.06
gamma_GTP	0.419859	0.014914	0.272427	0.406772	0.408744	0.049675	0.06
DRK_YN	0.377456	-0.278262	0.375580	0.268724	0.124624	0.151138	0.15

```
1 df_corr_1 = df_corr[abs(df_corr) > 0.2]
```

```
1 sns.heatmap(df_corr_1)
```

```
<Axes: >
```






```
1 input_feature = ['sex','age','height','weight','hemoglobin','gamma_GTP']
```

```
1 #df.drop('age',axis=1,inplace=True)
```

```
1 df.dtypes
```

```
sex                 float64
age                 float64
height              float64
weight              float64
waistline           float64
sight_left          float64
sight_right         float64
SBP                 float64
DBP                 float64
BLDS                float64
tot_chole           float64
HDL_chole           float64
LDL_chole           float64
triglyceride        float64
hemoglobin          float64
serum_creatinine    float64
SGOT_AST            float64
SGOT_ALT            float64
gamma_GTP           float64
DRK_YN              float64
dtype: object
```

```
1 # x=df.iloc[:,:-1].values
2 # x
```

```
1 new_x = df[input_feature]
2 new_x.head()
3 x = new_x.values
4 x
```

```
array([[  1. ,  35. , 170. ,  75. ,  17.1,  40. ],
       [  1. ,  30. , 180. ,  80. ,  15.8,  27. ],
       [  1. ,  40. , 165. ,  75. ,  15.8,  68. ],
       ...,
       [  1. ,  40. , 170. ,  75. ,  17.2,  76. ],
       [  0. ,  30. , 150. ,  55. ,  14.7,  24. ],
       [  1. ,  50. , 165. ,  70. ,  15. ,  76. ]])
```

```
1 x.shape
```

```
(23999, 6)
```

```
1 x.ndim
```

```
2
```

```
1 y=df.iloc[:,-1].values
2 y
```

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

```
1 y.ndim
```

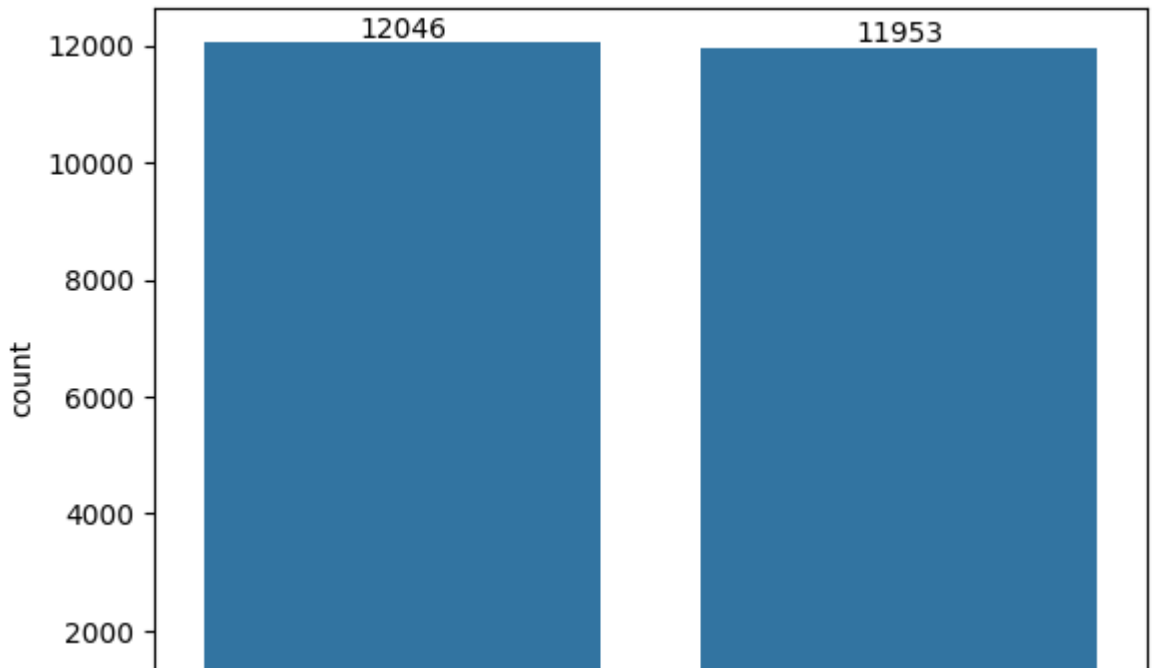
```
1
```

```
1 df['DRK_YN'].value_counts()
```

```
DRK_YN
0.0    12046
1.0    11953
Name: count, dtype: int64
```

Countplot of output

```
1 ax =sns.countplot(x=y,data=df)
2 for p in ax.patches:
3     ax.annotate(format(p.get_height(), '.0f'),
4                 (p.get_x() + p.get_width() / 2., p.get_height()),
5                 ha = 'center', va = 'center',
6                 xytext = (0, 5),
7                 textcoords = 'offset points')
```



Splitig of training & testing data

```
1 from sklearn.model_selection import train_test_split
2 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
3 x_train
```

```
array([[  1. ,  45. , 175. ,  70. ,  15.9,  22. ],
       [  0. ,  30. , 165. ,  50. ,  13.9,  15. ],
       [  0. ,  65. , 145. ,  45. ,  13.4,  14. ],
       ...,
       [  1. ,  45. , 175. ,  75. ,  14.7,  36. ],
       [  1. ,  40. , 170. ,  92.5,  16.1,  43. ],
       [  1. ,  50. , 165. ,  65. ,  16. ,  76. ]])
```

```
1 x_test
```

```
array([[  0. ,  45. , 160. ,  60. ,  14.4,  11. ],
       [  1. ,  65. , 160. ,  50. ,  12.8,  15. ],
       [  0. ,  60. , 140. ,  45. ,  11.8,  16. ],
       ...,
       [  0. ,  55. , 145. ,  50. ,  13.8,  21. ],
       [  0. ,  40. , 165. ,  55. ,  10.6,  10. ],
       [  0. ,  40. , 150. ,  45. ,  14.2,  15. ]])
```

```
1 y_train
```

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

```
1 y_test
```

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

```
1 pd.unique(y_train)
```

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

Normalization

```
1 from sklearn.preprocessing import StandardScaler
2 scaler=StandardScaler()
3 scaler.fit(x_train)
4 x_train=scaler.transform(x_train)
5 x_test=scaler.transform(x_test)
```

Model Creation

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report,Confusi
knn=KNeighborsClassifier(n_neighbors=7)
base=GaussianNB()
model=SVC()
lst=[knn,base,model]
```

Performance Evaluation

```python
for i in lst:
  i.fit(x_train,y_train)
  y_pred=i.predict(x_test)
  cmd=ConfusionMatrixDisplay(confusion_matrix(y_test,y_pred),display_labels=pd.unique(y_t
  print('\n')
  print('Model is ',i)
  print('\n')
  print(y_pred)
  print('!'*100)
  print('score is',accuracy_score(y_test,y_pred))
  print('*'*100)
  print('confusion is',confusion_matrix(y_test,y_pred))
  print('#'*100)
  print('classification_rp',classification_report(y_test,y_pred))
  print('CM display is',cmd.plot())
```

```
Model is  KNeighborsClassifier(n_neighbors=7)


[0. 0. 0. ... 0. 0. 1.]
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
score is 0.6841666666666667
****************************************************************************************
confusion is [[2405 1203]
 [1071 2521]]
########################################################################################
classification_rp               precision    recall  f1-score   support

         0.0       0.69      0.67      0.68      3608
         1.0       0.68      0.70      0.69      3592

    accuracy                           0.68      7200
   macro avg       0.68      0.68      0.68      7200
weighted avg       0.68      0.68      0.68      7200

CM display is <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0


Model is  GaussianNB()
```

```
[0. 0. 0. ... 0. 0. 0.]
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
score is 0.6944444444444444
******************************************************************************************
confusion is [[2526 1082]
 [1118 2474]]
##########################################################################################
classification_rp               precision    recall  f1-score   support

         0.0       0.69      0.70      0.70      3608
         1.0       0.70      0.69      0.69      3592

    accuracy                           0.69      7200
   macro avg       0.69      0.69      0.69      7200
weighted avg       0.69      0.69      0.69      7200

CM display is <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 6


Model is  SVC()


[0. 0. 0. ... 0. 0. 0.]
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
score is 0.7072222222222222
******************************************************************************************
confusion is [[2437 1171]
 [ 937 2655]]
##########################################################################################
classification_rp               precision    recall  f1-score   support
```

Model creation : Random Forest

```
         1.0       0.69      0.74      0.72      3592
```

```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=5, criterion='gini', random_state=0)
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
# from sklearn.svm import SVC
# model=SVC()
# model.fit(x_train,y_train)
# y_pred=model.predict(x_test)
# y_pred
```

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

Performnace evaluation

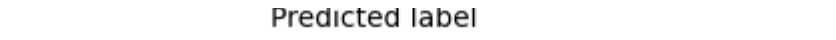
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report,Confusi
cm=confusion_matrix(y_test,y_pred)
cm
```

```
array([[2354, 1254],
```

```
[1245, 2347]])
```

```
1 ac=accuracy_score(y_test,y_pred)
2 ac
```

```
0.6529166666666667
```




```
1 cr=classification_report (y_test,y_pred)
2 print(cr)
```

```
                precision    recall  f1-score   support

           0.0       0.65      0.65      0.65      3608
           1.0       0.65      0.65      0.65      3592

      accuracy                           0.65      7200
     macro avg       0.65      0.65      0.65      7200
  weighted avg       0.65      0.65      0.65      7200
```

Model Creation : XGBclassifier

```
1 from xgboost import XGBClassifier
2 xgmodel = XGBClassifier(random_state=0)
3 xgmodel.fit(x_train,y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=0, ...)
```

```
1 y_pred =xgmodel.predict(x_test)
```

```
1 ac=accuracy_score(y_test,y_pred)
2 ac
```

```
0.7045833333333333
```

```
1 from sklearn.metrics import classification_report
2 print(classification_report(y_test,y_pred))
```

```
                precision    recall  f1-score   support
```

```
           0.0       0.72      0.68      0.70      3608
           1.0       0.69      0.73      0.71      3592

      accuracy                           0.70      7200
     macro avg       0.71      0.70      0.70      7200
  weighted avg       0.71      0.70      0.70      7200
```

Hyper Parameter Tuning XGBClassifier

```
# For Hyperparameter tuning
from sklearn.model_selection import GridSearchCV
xgb = XGBClassifier()

# Define the grid of hyperparameters to search
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 4, 5],
    'learning_rate': [0.05, 0.1, 0.2],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0]
}

# Instantiate GridSearchCV
grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=5, scoring='accuracy'

# Fit the grid search to the data
grid_search.fit(x_train, y_train)

# Get the best parameters and the best score
best_params = grid_search.best_params_
print("Best Parameters:", best_params)
```

```
Best Parameters: {'colsample_bytree': 0.9, 'learning_rate': 0.05, 'max_depth': 3, 'n_est
```

```
xgb1 = XGBClassifier(colsample_bytree=0.9,learning_rate=0.05,max_depth=3,n_estimators=200
xgb1.fit(x_train,y_train)
y_pred1 = xgb1.predict(x_test)
y_pred1
```

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

```
score = accuracy_score(y_test,y_pred1)
score
```

```
0.7095833333333333
```

```
pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.2.5-cp310-cp310-manylinux2014_x86_64.whl (98.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.2/98.2 MB 2.3 MB/s eta 0:00:00
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from ca
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from c
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catb
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-package
Installing collected packages: catboost
Successfully installed catboost-1.2.5
```

Model Creation : Catbooster

```
1 from catboost import CatBoostClassifier
2 cb = CatBoostClassifier()
3 cb.fit(x_train,y_train)
4 y_pred=cb.predict(x_test)
5 y_pred
```

```
Learning rate set to 0.034366
0:      learn: 0.6832799        total: 51.7ms   remaining: 51.6s
1:      learn: 0.6743027        total: 56.9ms   remaining: 28.4s
2:      learn: 0.6666223        total: 60.6ms   remaining: 20.1s
3:      learn: 0.6588005        total: 65.5ms   remaining: 16.3s
4:      learn: 0.6516208        total: 69.5ms   remaining: 13.8s
5:      learn: 0.6453102        total: 74.3ms   remaining: 12.3s
6:      learn: 0.6399327        total: 78.1ms   remaining: 11.1s
7:      learn: 0.6346929        total: 81.6ms   remaining: 10.1s
8:      learn: 0.6293743        total: 86.8ms   remaining: 9.55s
9:      learn: 0.6239359        total: 91.7ms   remaining: 9.08s
10:     learn: 0.6191680        total: 97ms     remaining: 8.72s
11:     learn: 0.6148740        total: 102ms    remaining: 8.41s
12:     learn: 0.6109272        total: 107ms    remaining: 8.15s
13:     learn: 0.6068740        total: 114ms    remaining: 8.05s
14:     learn: 0.6033009        total: 119ms    remaining: 7.85s
15:     learn: 0.5999638        total: 124ms    remaining: 7.63s
16:     learn: 0.5971658        total: 129ms    remaining: 7.47s
17:     learn: 0.5945976        total: 134ms    remaining: 7.33s
18:     learn: 0.5926530        total: 139ms    remaining: 7.18s
19:     learn: 0.5901456        total: 144ms    remaining: 7.07s
20:     learn: 0.5878390        total: 150ms    remaining: 6.97s
21:     learn: 0.5855610        total: 155ms    remaining: 6.88s
22:     learn: 0.5835230        total: 160ms    remaining: 6.79s
23:     learn: 0.5817418        total: 165ms    remaining: 6.73s
24:     learn: 0.5798406        total: 171ms    remaining: 6.67s
25:     learn: 0.5782070        total: 175ms    remaining: 6.56s
26:     learn: 0.5766043        total: 181ms    remaining: 6.51s
```

```
27:     learn: 0.5752667        total: 186ms    remaining: 6.46s
28:     learn: 0.5738937        total: 191ms    remaining: 6.41s
29:     learn: 0.5724349        total: 200ms    remaining: 6.48s
30:     learn: 0.5712017        total: 206ms    remaining: 6.45s
31:     learn: 0.5700903        total: 212ms    remaining: 6.4s
32:     learn: 0.5690052        total: 217ms    remaining: 6.36s
33:     learn: 0.5680874        total: 222ms    remaining: 6.31s
34:     learn: 0.5672090        total: 228ms    remaining: 6.28s
35:     learn: 0.5662916        total: 233ms    remaining: 6.23s
36:     learn: 0.5655308        total: 238ms    remaining: 6.19s
37:     learn: 0.5646765        total: 245ms    remaining: 6.19s
38:     learn: 0.5641019        total: 248ms    remaining: 6.12s
39:     learn: 0.5635243        total: 252ms    remaining: 6.05s
40:     learn: 0.5628528        total: 258ms    remaining: 6.02s
41:     learn: 0.5622065        total: 263ms    remaining: 6s
42:     learn: 0.5615185        total: 268ms    remaining: 5.97s
43:     learn: 0.5610247        total: 274ms    remaining: 5.95s
44:     learn: 0.5604456        total: 279ms    remaining: 5.92s
45:     learn: 0.5599715        total: 284ms    remaining: 5.89s
46:     learn: 0.5594971        total: 290ms    remaining: 5.87s
47:     learn: 0.5590498        total: 295ms    remaining: 5.84s
48:     learn: 0.5585199        total: 300ms    remaining: 5.82s
49:     learn: 0.5579803        total: 305ms    remaining: 5.8s
50:     learn: 0.5575636        total: 310ms    remaining: 5.78s
51:     learn: 0.5571564        total: 316ms    remaining: 5.76s
52:     learn: 0.5567077        total: 321ms    remaining: 5.74s
53:     learn: 0.5563491        total: 326ms    remaining: 5.72s
54:     learn: 0.5560238        total: 331ms    remaining: 5.7s
55:     learn: 0.5557000        total: 337ms    remaining: 5.68s
56:     learn: 0.5554010        total: 342ms    remaining: 5.66s
```

```
1 ac=accuracy_score(y_test,y_pred)
2 ac
```

```
0.7109722222222222
```

```
1 input_feature
```

```
['sex', 'age', 'height', 'weight', 'hemoglobin', 'gamma_GTP']
```

```
1 df1 = pd.DataFrame(x_test)
2 df1
```

	0	1	2	3	4	5
0	-1.062996	-0.189922	-0.244224	-0.264138	0.105143	-0.980150
1	0.940737	1.221025	-0.244224	-1.087465	-0.926047	-0.782980
2	-1.062996	0.868288	-2.400426	-1.499128	-1.570540	-0.733688
3	0.940737	-0.189922	1.372929	-0.675801	0.814086	0.202868
4	-1.062996	0.162815	-0.244224	-1.087465	-1.183844	-0.930857
...	...	...	...	...	...	...
7195	-1.062996	-1.600870	-0.783274	-0.675801	-0.539351	-0.832273

7196	0.940737	-0.542659	1.372929	1.382516	-0.539351	0.104283
7197	-1.062996	0.515551	-1.861376	-1.087465	-0.281553	-0.487226
7198	-1.062996	-0.542659	0.294827	-0.675801	-2.343933	-1.029442
7199	-1.062996	-0.542659	-1.322325	-1.499128	-0.023756	-0.782980

7200 rows × 6 columns

```
1 df2 = pd.DataFrame(y_test)
2 df2
```

	0
0	1.0
1	0.0
2	0.0
3	1.0
4	0.0
...	...
7195	1.0
7196	1.0
7197	0.0
7198	0.0
7199	0.0

7200 rows × 1 columns

```
1 df3 = pd.concat([df1,df2],axis=1)
2 df3.columns = [0,1,2,3,4,5,'out']
3 df3
```

	0	1	2	3	4	5	out
0	-1.062996	-0.189922	-0.244224	-0.264138	0.105143	-0.980150	1.0
1	0.940737	1.221025	-0.244224	-1.087465	-0.926047	-0.782980	0.0
2	-1.062996	0.868288	-2.400426	-1.499128	-1.570540	-0.733688	0.0
3	0.940737	-0.189922	1.372929	-0.675801	0.814086	0.202868	1.0
4	-1.062996	0.162815	-0.244224	-1.087465	-1.183844	-0.930857	0.0
...	...	...	...	...	...	...	...

7195	-1.062996	-1.600870	-0.783274	-0.675801	-0.539351	-0.832273	1.0
7196	0.940737	-0.542659	1.372929	1.382516	-0.539351	0.104283	1.0
7197	-1.062996	0.515551	-1.861376	-1.087465	-0.281553	-0.487226	0.0
7198	-1.062996	-0.542659	0.294827	-0.675801	-2.343933	-1.029442	0.0
7199	-1.062996	-0.542659	-1.322325	-1.499128	-0.023756	-0.782980	0.0

7200 rows × 7 columns

```
1 drinker_data = df3[df3.out == 1]
2 drinker_data
```

	0	1	2	3	4	5	out
0	-1.062996	-0.189922	-0.244224	-0.264138	0.105143	-0.980150	1.0
3	0.940737	-0.189922	1.372929	-0.675801	0.814086	0.202868	1.0
6	0.940737	1.221025	0.294827	0.147526	0.105143	-0.339349	1.0
7	0.940737	-0.895396	1.372929	-0.264138	0.620738	0.892962	1.0
9	0.940737	-1.953606	0.833878	0.970853	-1.312743	-0.437933	1.0
...	...	...	...	...	...	...	...
7191	0.940737	0.868288	0.294827	0.147526	0.234042	2.223856	1.0
7192	0.940737	-0.542659	1.372929	1.382516	1.651928	-0.043594	1.0
7193	-1.062996	0.868288	-1.322325	-0.675801	-0.926047	0.597207	1.0
7195	-1.062996	-1.600870	-0.783274	-0.675801	-0.539351	-0.832273	1.0
7196	0.940737	-0.542659	1.372929	1.382516	-0.539351	0.104283	1.0

3592 rows × 7 columns

```
1 non_drinker_data = df3[df3.out == 0]
2 non_drinker_data
```

	0	1	2	3	4	5	out
1	0.940737	1.221025	-0.244224	-1.087465	-0.926047	-0.782980	0.0
2	-1.062996	0.868288	-2.400426	-1.499128	-1.570540	-0.733688	0.0
4	-1.062996	0.162815	-0.244224	-1.087465	-1.183844	-0.930857	0.0
5	0.940737	0.162815	0.833878	0.970853	1.136333	0.202868	0.0
8	-1.062996	-0.189922	-0.783274	-1.499128	1.071884	-0.782980	0.0
...	...	...	...	...	...	...	...

7188	0.940737	0.515551	-0.783274	-0.675801	0.427390	0.054991	0.0
7194	-1.062996	-0.895396	-0.783274	-1.087465	-1.119395	-0.782980	0.0
7197	-1.062996	0.515551	-1.861376	-1.087465	-0.281553	-0.487226	0.0
7198	-1.062996	-0.542659	0.294827	-0.675801	-2.343933	-1.029442	0.0
7199	-1.062996	-0.542659	-1.322325	-1.499128	-0.023756	-0.782980	0.0

3608 rows × 7 columns

```
1 cb.predict([[0.940737,-0.542659,1.372929,1.382516,-0.539351,0.104283]])
```

```
array([1.])
```