In [1]:

```python
import pandas as pd
A=pd.read_csv(r'C:\Users\Admin\Downloads\Chronic_Kidney_Dsease_data.csv
A
```

Out[1]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
0	1	71	0	0	0	2	31.069414	
1	2	34	0	0	1	3	29.692119	
2	3	80	1	1	0	1	37.394822	
3	4	40	0	2	0	1	31.329680	
4	5	43	0	1	1	2	23.726311	
...	...	...	...	...	...	...	...	
1654	1655	90	0	0	1	2	39.677059	
1655	1656	34	0	0	2	1	28.922015	
1656	1657	84	0	0	2	3	21.951219	
1657	1658	90	0	0	2	2	24.964149	
1658	1659	34	1	1	0	0	19.253258	

1659 rows × 54 columns

In [2]:

```python
pd.set_option("Display.max_columns",100)
```

In [3]:

```python
A
```

Out[3]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
0	1	71	0	0	0	2	31.069414	
1	2	34	0	0	1	3	29.692119	
2	3	80	1	1	0	1	37.394822	
3	4	40	0	2	0	1	31.329680	
4	5	43	0	1	1	2	23.726311	
...	...	...	...	...	...	...	...	
1654	1655	90	0	0	1	2	39.677059	
1655	1656	34	0	0	2	1	28.922015	
1656	1657	84	0	0	2	3	21.951219	
1657	1658	90	0	0	2	2	24.964149	
1658	1659	34	1	1	0	0	19.253258	

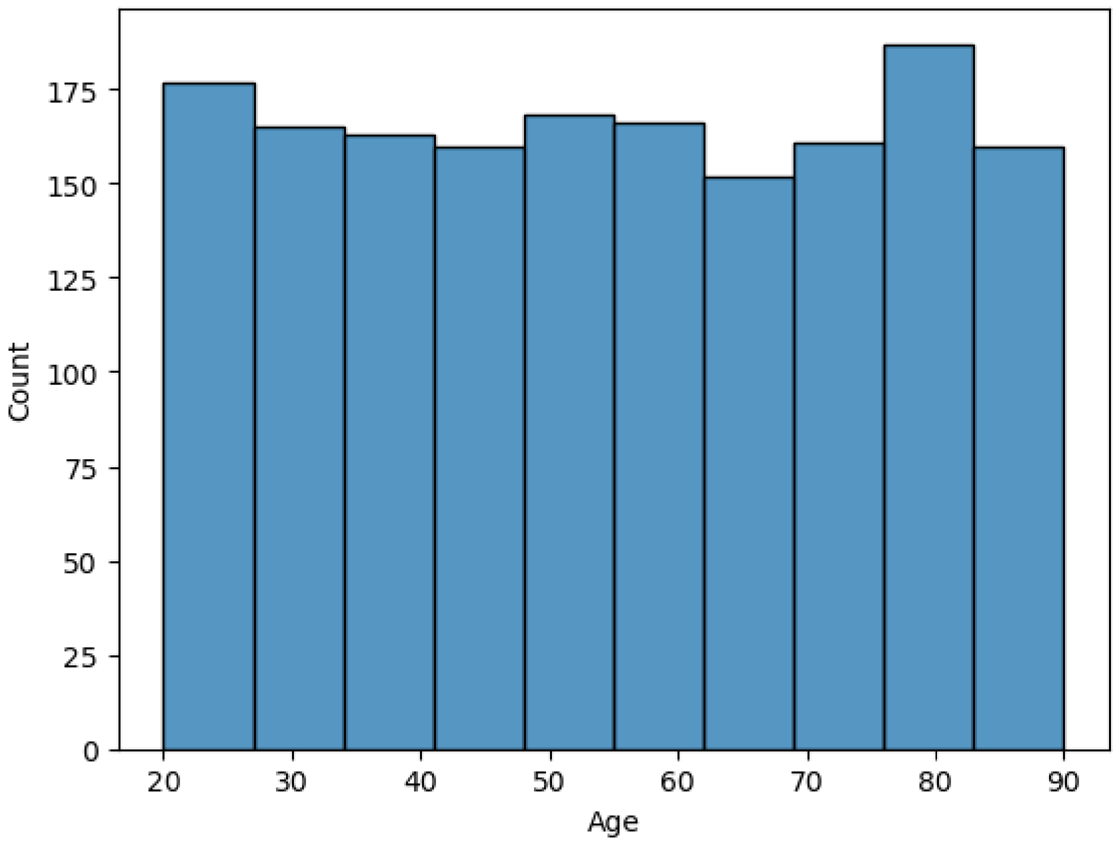
1659 rows × 54 columns

In [4]:
```
import seaborn as sns
sns.histplot(A['Age'],bins=10)

```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Futur
eWarning: use_inf_as_na option is deprecated and will be removed in a futu
re version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[4]:
```
<Axes: xlabel='Age', ylabel='Count'>
```



In [6]:
```
A.loc[(A['FatigueLevels']>=5)&(A['Diagnosis']==1)]
```

Out[6]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	Sı
1	2	34	0	0	1	3	29.692119	
3	4	40	0	2	0	1	31.329680	
5	6	22	0	0	0	1	39.155643	
6	7	41	0	1	0	1	35.040487	
15	16	52	1	0	0	2	25.059839	
...	...	...	...	...	...	...	...	
1642	1643	60	1	0	1	2	34.923694	
1643	1644	67	0	0	1	2	16.572316	
1652	1653	20	0	0	1	3	20.378015	
1653	1654	73	1	0	1	3	35.634449	
1658	1659	34	1	1	0	0	19.253258	

773 rows × 54 columns

In [7]:
```
A.drop(columns="DoctorInCharge",inplace=True)
```

In [8]:
```
A
```

Out[8]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	Sı
0	1	71	0	0	0	2	31.069414	
1	2	34	0	0	1	3	29.692119	
2	3	80	1	1	0	1	37.394822	
3	4	40	0	2	0	1	31.329680	
4	5	43	0	1	1	2	23.726311	
...	...	...	...	...	...	...	...	
1654	1655	90	0	0	1	2	39.677059	
1655	1656	34	0	0	2	1	28.922015	
1656	1657	84	0	0	2	3	21.951219	
1657	1658	90	0	0	2	2	24.964149	
1658	1659	34	1	1	0	0	19.253258	

1659 rows × 53 columns

In [9]:
```
c=A[['SystolicBP','DiastolicBP','FastingBloodSugar','ProteinInUrine','(
c
```

Out[9]:

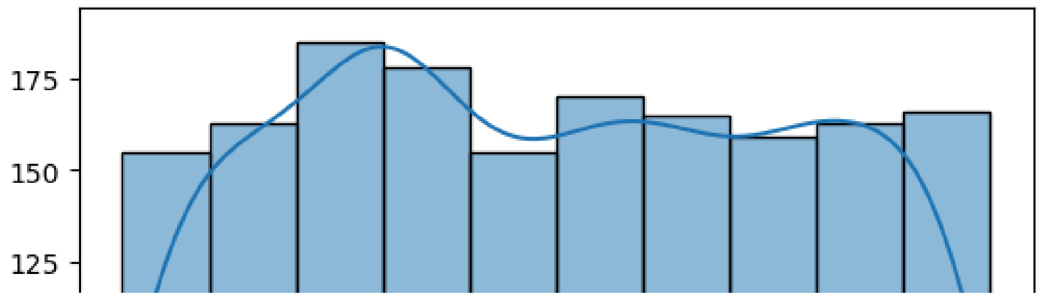
	SystolicBP	DiastolicBP	FastingBloodSugar	ProteinInUrine	CholesterolTotal	Cholester
0	113	83	72.510788	0.744980	207.728670	85.8
1	120	67	100.848875	3.052317	189.450727	86.3
2	147	106	160.989441	1.157839	284.137622	132.2
3	117	65	188.506620	3.745871	235.112124	93.4
4	98	66	82.156699	2.570993	258.277566	171.7
...	...	...	...	...	...	
1654	130	89	195.433613	2.926489	184.518899	133.1
1655	127	103	73.606489	3.496617	253.709988	131.9
1656	118	102	163.084321	3.549633	221.399305	183.3
1657	163	87	98.794331	3.816679	261.911664	184.3
1658	111	89	161.181060	0.335946	174.746532	123.0

1659 rows × 9 columns

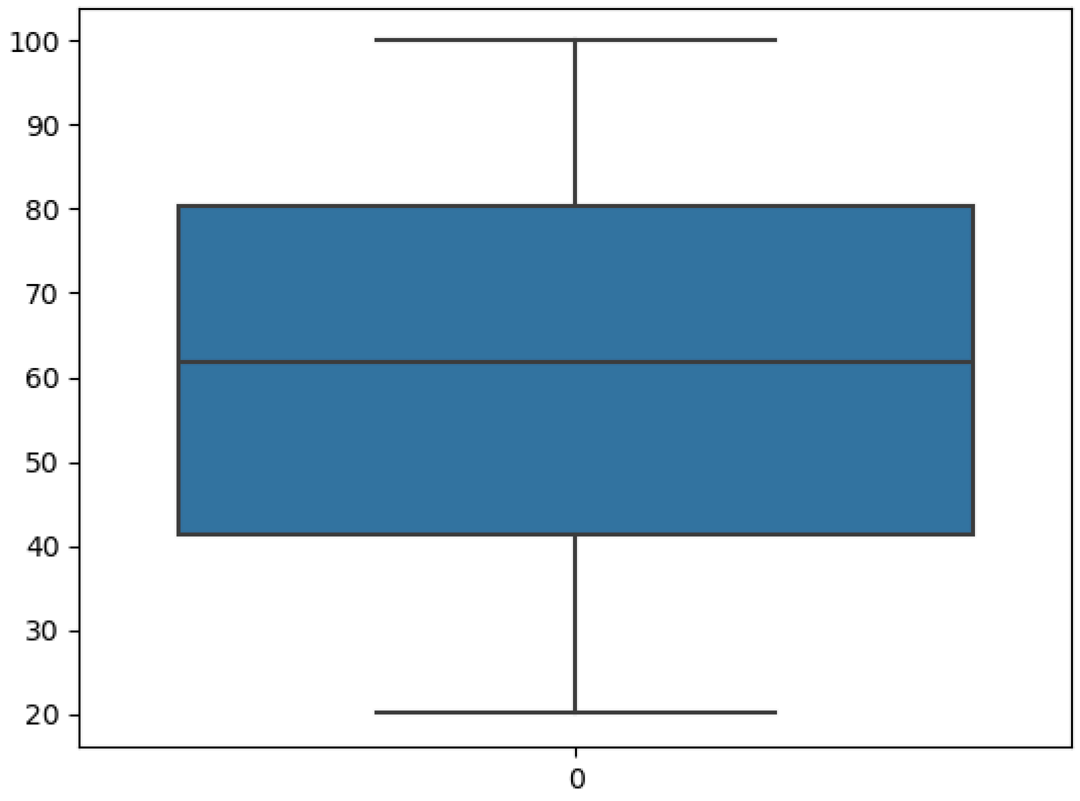
In [10]:
```
import matplotlib.pyplot as plt
import seaborn as sns
for i in c.columns:
    sns.histplot(c[i],bins=10,edgecolor='black',kde=True)
    print("mean",(c[i].mean()))
    print("median",(c[i].median()))
    print("mode",(c[i].mode()[0]))
    print("min",(c[i].min()))
    print("max",(c[i].max()))
    plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in
a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):

mean 134.39240506329114
median 134.0
mode 116
min 90
max 179
```

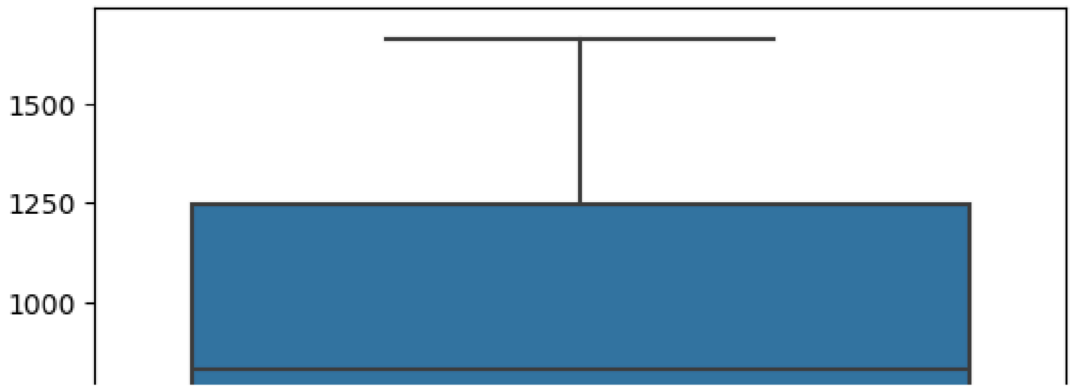


In [11]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(A['CholesterolHDL'])
plt.show()
```



In [12]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
for i in A.columns:
    sns.boxplot(A[i])
    q1=A[i].quantile(0.25)
    q2=A[i].quantile(0.5)
    q3=A[i].quantile(0.75)
    iqr=q3-q1
    low=q1-(1.5*iqr)
    high=q3+(1.5*iqr)
    print(i.upper())
    print("Q1",q1)
    print("Q2",q2)
    print("Q3",q3)
    print("IQR",iqr)
    print("LOW",low)
    print("High",high)
    plt.show()
```

```
PATIENTID
Q1 415.5
Q2 830.0
Q3 1244.5
IQR 829.0
LOW -828.0
High 2488.0
```



In [13]:
```
A
```

Out[13]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	Sı
0	1	71	0	0	0	2	31.069414	
1	2	34	0	0	1	3	29.692119	
2	3	80	1	1	0	1	37.394822	
3	4	40	0	2	0	1	31.329680	
4	5	43	0	1	1	2	23.726311	
...	...	...	...	...	...	...	...	
1654	1655	90	0	0	1	2	39.677059	
1655	1656	34	0	0	2	1	28.922015	
1656	1657	84	0	0	2	3	21.951219	
1657	1658	90	0	0	2	2	24.964149	
1658	1659	34	1	1	0	0	19.253258	

1659 rows × 53 columns

In [14]:
```
A['GFR'].max()
```

Out[14]:
```
119.9202609235662
```

In [15]:
```
A['ACR'].max()
```

Out[15]:
```
299.58001918489805
```

In [16]:
```
B=A.loc[(A['GFR'].between(15,30))&(A['ACR']>=30)&(A['FastingBloodSugar
B
```

Out[16]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
8	9	21	0	1	0	2	22.323130	
33	34	40	0	0	0	3	31.539777	
38	39	33	0	1	0	0	19.709625	
73	74	64	1	0	1	3	36.391092	
103	104	33	0	0	0	1	18.214489	
...	...	...	...	...	...	...	...	
1600	1601	58	0	0	1	1	22.454872	
1604	1605	42	1	0	1	2	38.555843	
1606	1607	57	0	1	0	2	23.126441	
1656	1657	84	0	0	2	3	21.951219	
1658	1659	34	1	1	0	0	19.253258	

111 rows × 53 columns

In [17]:
```
Z=A.loc[A['Age']<=40]
Z
```

Out[17]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
1	2	34	0	0	1	3	29.692119	
3	4	40	0	2	0	1	31.329680	
5	6	22	0	0	0	1	39.155643	
8	9	21	0	1	0	2	22.323130	
11	12	21	1	2	0	3	16.799520	
...	...	...	...	...	...	...	...	
1648	1649	38	0	1	2	3	17.863632	
1650	1651	32	0	0	1	3	35.253136	
1652	1653	20	0	0	1	3	20.378015	
1655	1656	34	0	0	2	1	28.922015	
1658	1659	34	1	1	0	0	19.253258	

505 rows × 53 columns

In [18]:
```
Z.loc[(Z['CholesterolTotal']>=200)]
```

Out[18]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI
3	4	40	0	2	0	1	31.329680
5	6	22	0	0	0	1	39.155643
8	9	21	0	1	0	2	22.323130
11	12	21	1	2	0	3	16.799520
14	15	40	0	2	0	3	27.000463
...	...	...	...	...	...	...	...
1644	1645	32	0	0	1	3	32.164663
1647	1648	23	0	3	0	1	39.109585
1648	1649	38	0	1	2	3	17.863632
1652	1653	20	0	0	1	3	20.378015
1655	1656	34	0	0	2	1	28.922015

334 rows × 53 columns

In [19]:
```
Z.loc[Z['CholesterolLDL']>=160]
```

Out[19]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI
35	36	37	0	0	1	2	17.446425
36	37	23	1	0	1	3	29.403983
45	46	27	0	0	1	1	39.014213
52	53	25	0	0	1	1	18.073047
57	58	37	1	0	0	2	31.143982
...	...	...	...	...	...	...	...
1578	1579	25	0	0	2	0	20.606460
1581	1582	21	1	0	1	1	29.004297
1583	1584	39	1	0	2	1	34.835915
1593	1594	31	0	2	1	2	16.522806
1650	1651	32	0	0	1	3	35.253136

123 rows × 53 columns

In [20]:
```
Z['Smoking'].value_counts()
```

Out[20]:
```
Smoking
0    357
1    148
Name: count, dtype: int64
```

In [21]:
```
Z['Diagnosis'].value_counts()
```

Out[21]:
```
Diagnosis
1    468
0     37
Name: count, dtype: int64
```

In [22]:
```
U=A.loc[A['Age'].between(40,60)]
U
```

Out[22]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
3	4	40	0	2	0	1	31.329680	
4	5	43	0	1	1	2	23.726311	
6	7	41	0	1	0	1	35.040487	
9	10	49	0	3	0	1	24.338507	
10	11	57	1	1	0	3	31.749248	
...	...	...	...	...	...	...	...	
1631	1632	50	1	0	2	0	33.037229	
1641	1642	52	0	3	1	2	18.688796	
1642	1643	60	1	0	1	2	34.923694	
1645	1646	50	0	2	0	2	24.161423	
1651	1652	42	1	1	1	3	21.653960	

491 rows × 53 columns

In [23]:
```
U.loc[U['CholesterolTotal']>=220]
```

Out[23]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
3	4	40	0	2	0	1	31.329680	
4	5	43	0	1	1	2	23.726311	
6	7	41	0	1	0	1	35.040487	
10	11	57	1	1	0	3	31.749248	
15	16	52	1	0	0	2	25.059839	
...	...	...	...	...	...	...	...	
1607	1608	42	1	0	0	0	18.953164	
1614	1615	48	1	0	0	1	17.789558	
1628	1629	52	0	2	0	2	35.979809	
1641	1642	52	0	3	1	2	18.688796	
1645	1646	50	0	2	0	2	24.161423	

268 rows × 53 columns

In [24]:
```
U['Diagnosis'].value_counts()
```

Out[24]:
```
Diagnosis
1    445
0     46
Name: count, dtype: int64
```

In [25]:
```
U['Smoking'].value_counts()
```

Out[25]:
```
Smoking
0    361
1    130
Name: count, dtype: int64
```

In [26]:
```
O=A.loc[A['Age']>=60]
O
```

Out[26]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	Sı
0	1	71	0	0	0	2	31.069414	
2	3	80	1	1	0	1	37.394822	
7	8	72	1	0	1	3	30.760440	
12	13	83	1	0	0	2	21.417015	
13	14	79	0	1	1	1	32.847523	
...	...	...	...	...	...	...	...	
1649	1650	89	0	0	1	2	22.859626	
1653	1654	73	1	0	1	3	35.634449	
1654	1655	90	0	0	1	2	39.677059	
1656	1657	84	0	0	2	3	21.951219	
1657	1658	90	0	0	2	2	24.964149	

697 rows × 53 columns

In [27]:
```
O['Smoking'].value_counts()
```

Out[27]:
```
Smoking
0    480
1    217
Name: count, dtype: int64
```

In [ ]:
```

```

In [28]:
```
O['Diagnosis'].value_counts()
```

Out[28]:
```
Diagnosis
1    642
0     55
Name: count, dtype: int64
```

In [29]:
```
A.loc[A['GFR'].between(15,30)]
```

Out[29]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
8	9	21	0	1	0	2	22.323130	
16	17	77	1	2	1	2	18.786441	
24	25	81	1	3	0	0	17.998046	
27	28	70	1	1	2	1	20.814202	
28	29	74	1	0	1	2	30.192398	
...	...	...	...	...	...	...	...	
1624	1625	27	1	1	1	3	30.096848	
1630	1631	38	1	0	0	2	27.556872	
1652	1653	20	0	0	1	3	20.378015	
1656	1657	84	0	0	2	3	21.951219	
1658	1659	34	1	1	0	0	19.253258	

245 rows × 53 columns

In [30]:
```
c=A.loc[(A['GFR']<=80)&(A['ACR']>30)&(A['FastingBloodSugar']>=126)]
c
```

Out[30]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
3	4	40	0	2	0	1	31.329680	
5	6	22	0	0	0	1	39.155643	
6	7	41	0	1	0	1	35.040487	
8	9	21	0	1	0	2	22.323130	
10	11	57	1	1	0	3	31.749248	
...	...	...	...	...	...	...	...	
1637	1638	85	1	0	1	1	25.521931	
1644	1645	32	0	0	1	3	32.164663	
1645	1646	50	0	2	0	2	24.161423	
1656	1657	84	0	0	2	3	21.951219	
1658	1659	34	1	1	0	0	19.253258	

470 rows × 53 columns

In [31]:
```
c['Diagnosis'].value_counts()     #with above Values of GFR & ACR large
```

Out[31]:
```
Diagnosis
1    446
0     24
Name: count, dtype: int64
```

In [32]:
```
B.corr()
```

Out[32]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicSta
PatientID	1.000000	0.281177	-0.063150	-0.127865	0.099
Age	0.281177	1.000000	-0.009352	0.004132	0.122
Gender	-0.063150	-0.009352	1.000000	-0.126886	0.010
Ethnicity	-0.127865	0.004132	-0.126886	1.000000	-0.085
SocioeconomicStatus	0.099698	0.122650	0.010155	-0.085799	1.000
EducationLevel	0.126318	-0.031194	-0.081778	0.164633	0.030
BMI	-0.095005	0.029355	0.150483	-0.064117	-0.016
Smoking	-0.033446	0.028314	0.081155	0.031241	-0.016
AlcoholConsumption	0.170191	0.073982	-0.279148	0.086042	0.026
PhysicalActivity	0.004639	0.049942	-0.034800	-0.174241	-0.026
DietQuality	0.006130	-0.045474	-0.012971	-0.016591	0.107
SleepQuality	0.047131	0.258393	-0.061956	0.059864	0.122
FamilyHistoryKidneyDisease	0.089363	-0.033673	-0.043174	-0.113717	-0.109
FamilyHistoryHypertension	-0.073821	-0.102523	0.065415	-0.022132	0.007
FamilyHistoryDiabetes	-0.014308	0.071968	-0.123600	0.432549	0.058
PreviousAcuteKidneyInjury	-0.153130	-0.128514	0.080141	-0.008318	0.003
UrinaryTractInfections	0.070493	0.039371	-0.079056	-0.017572	-0.127
SystolicBP	-0.000519	0.071543	0.109796	-0.075831	0.120
DiastolicBP	-0.001133	0.130244	0.014882	-0.149683	0.015
FastingBloodSugar	-0.069279	0.036648	-0.054475	0.128897	-0.006
HbA1c	-0.167357	-0.051343	0.038618	0.151563	-0.141
SerumCreatinine	0.004171	-0.157618	0.040158	0.098403	-0.047
BUNLevels	-0.053555	0.170749	-0.041353	-0.043701	0.081
GFR	0.038924	0.130576	0.009629	0.128553	0.251
ProteinInUrine	-0.029651	-0.239414	-0.018400	-0.199619	-0.022
ACR	-0.041942	-0.025637	-0.078283	0.147787	-0.002
SerumElectrolytesSodium	0.154581	-0.034118	0.039299	0.036944	0.141
SerumElectrolytesPotassium	-0.004817	-0.060267	0.040936	0.079561	-0.118
SerumElectrolytesCalcium	-0.043522	0.039553	-0.085322	0.010801	-0.181
SerumElectrolytesPhosphorus	0.117359	0.041771	0.014910	0.003287	-0.153
HemoglobinLevels	0.235978	0.120928	-0.161849	0.056450	-0.005
CholesterolTotal	-0.002912	0.030816	0.092664	0.019986	-0.213
CholesterolLDL	0.129102	0.132162	0.118107	-0.100785	0.236
CholesterolHDL	-0.123807	-0.105286	0.116156	0.139784	0.017
CholesterolTriglycerides	0.108603	-0.047857	0.123429	0.022739	-0.024
ACEInhibitors	-0.077094	-0.004805	0.012460	-0.049275	-0.097
Diuretics	0.068552	0.015516	0.047429	-0.072217	-0.036
NSAIDsUse	-0.091169	0.021904	0.178713	0.111872	0.009

	PatientID	Age	Gender	Ethnicity	SocioeconomicSta
Statins	-0.044253	0.012992	-0.087938	-0.082742	-0.055
AntidiabeticMedications	0.117324	0.090226	0.051938	-0.076497	-0.205
Edema	-0.097884	-0.088768	0.054577	-0.103310	0.112
FatigueLevels	0.012330	-0.160864	-0.054272	-0.012493	-0.160
NauseaVomiting	-0.009781	-0.148326	0.132523	0.018036	0.075
MuscleCramps	0.035189	-0.061554	-0.006403	-0.055356	0.047
Itching	-0.180936	0.037806	0.079327	0.081945	-0.025
QualityOfLifeScore	-0.135282	-0.071487	-0.017731	0.146199	-0.003
HeavyMetalsExposure	0.075328	0.123791	0.087423	-0.040251	0.170
OccupationalExposureChemicals	-0.023389	-0.207247	0.024006	0.004008	0.004
WaterQuality	-0.070377	-0.119718	0.167282	-0.088152	0.005
MedicalCheckupsFrequency	-0.026703	-0.156582	-0.045760	0.148800	-0.249
MedicationAdherence	0.044617	0.078179	-0.174368	-0.029464	0.142
HealthLiteracy	-0.149891	0.054072	0.074078	0.035623	-0.130
Diagnosis	-0.297615	-0.098104	0.084649	0.097842	-0.060

In [33]:
```
C=B.loc[(B['Smoking']==0)&(B['AlcoholConsumption']>=10)]
C #people with high gfr,ACR & HIGH alcoholcon but not smoking have stag
```

Out[33]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI
137	138	22	0	0	1	1	20.403150
164	165	72	0	0	0	0	39.017869
193	194	58	0	0	0	1	35.328750
214	215	22	1	3	0	2	37.093335
222	223	44	1	3	1	2	23.442979
223	224	37	0	2	1	3	28.731712
256	257	80	1	0	1	2	17.631629
334	335	24	0	1	1	1	25.152750
344	345	38	0	1	1	3	18.218680
347	348	38	1	1	2	3	39.358736
367	368	45	1	3	1	3	33.154737
388	389	20	1	0	2	3	29.490811
393	394	21	1	0	0	1	19.234912
412	413	32	0	1	2	2	23.505406
436	437	31	1	2	0	1	35.162399
453	454	71	0	3	1	0	16.160841
500	501	58	0	1	2	2	38.348783
526	527	40	0	0	0	3	30.775854
597	598	64	1	0	2	0	35.638271
639	640	53	0	0	0	3	39.435574
713	714	86	1	0	0	2	35.526959
766	767	20	0	1	2	2	36.501838
854	855	85	0	0	2	1	34.531424
1071	1072	21	0	1	1	2	32.597836
1095	1096	81	1	3	1	2	26.759289
1156	1157	62	0	0	2	1	21.961632
1179	1180	81	0	0	0	3	16.310142
1191	1192	64	0	3	0	0	15.889554
1242	1243	51	1	0	0	1	19.416076
1249	1250	48	0	0	2	3	36.483320
1273	1274	77	0	2	2	3	27.195571
1324	1325	56	0	1	0	1	27.787026
1327	1328	39	1	1	1	3	36.113038
1353	1354	65	1	0	1	3	38.180162
1458	1459	51	0	0	1	2	19.923337
1467	1468	77	0	0	1	3	17.839114
1473	1474	33	0	0	2	2	15.657443
1501	1502	68	0	1	0	3	23.087194

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	Sı
1531	1532	61	0	3	1	2	18.418057	
1540	1541	79	0	0	1	1	34.499012	
1546	1547	85	0	2	2	3	23.807559	
1568	1569	76	1	0	2	0	35.889120	
1592	1593	70	1	0	0	2	15.181820	
1600	1601	58	0	0	1	1	22.454872	
1606	1607	57	0	1	0	2	23.126441	
1656	1657	84	0	0	2	3	21.951219	

In [34]:
```python
Dx=B.loc[(B['Smoking']==1)&(B['AlcoholConsumption']>=10)]
Dx
```

Out[34]:

HeavyMetalsExposure	OccupationalExposureChemicals	WaterQuality	MedicalCheckupsFreque
0	0	0	2.353
0	0	0	2.444
0	1	0	3.500
0	1	0	0.900
0	0	1	3.739
0	1	0	1.819
0	0	0	1.467
0	0	0	0.112
0	0	0	3.776
0	1	0	2.563
0	0	0	1.475
0	0	0	2.915
0	0	0	0.345
0	0	0	2.292
0	1	0	2.428
0	0	0	1.754

In [35]:
```python
D=B.loc[(B['Smoking']==1)&(B['AlcoholConsumption']<=10)]
D
```

Out[35]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
33	34	40	0	0	0	3	31.539777	
171	172	75	1	0	2	0	28.935732	
255	256	88	1	0	2	0	30.093607	
282	283	65	1	2	0	3	20.594460	
297	298	45	0	0	0	0	38.599898	
299	300	44	0	2	1	1	36.254299	
312	313	34	1	0	0	1	36.191008	
313	314	85	1	3	0	3	34.289243	
353	354	31	0	3	0	0	17.997201	
552	553	47	1	0	0	1	20.711789	
764	765	31	0	3	2	3	31.069342	
811	812	54	1	0	2	3	37.766075	
1111	1112	38	1	0	1	3	35.812468	
1270	1271	59	1	0	1	2	34.801115	
1271	1272	64	0	0	2	1	22.318539	
1296	1297	64	1	0	1	2	22.742049	
1335	1336	84	0	1	1	1	25.762340	
1604	1605	42	1	0	1	2	38.555843	

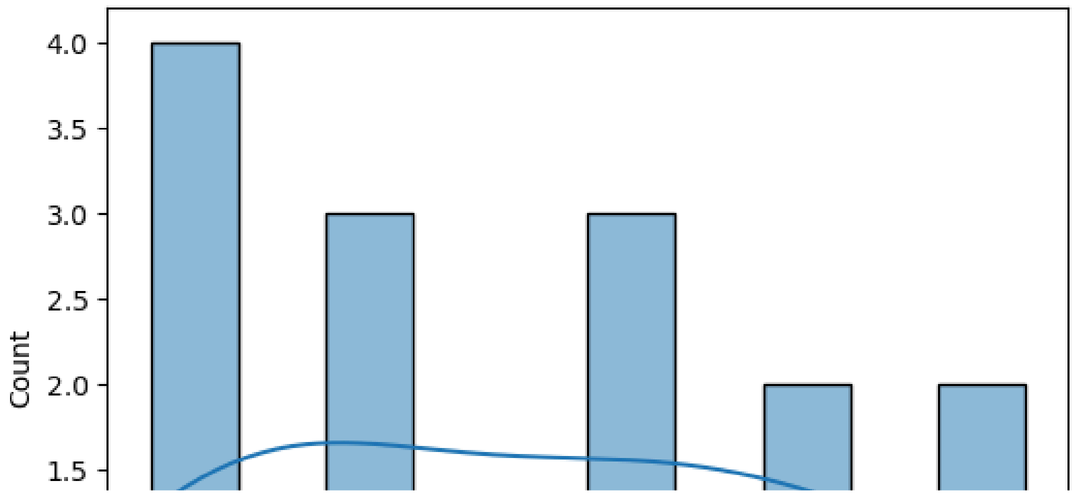
In [36]:
```
E=D[['SystolicBP','DiastolicBP','FastingBloodSugar','ProteinInUrine','(
E
```

Out[36]:

	SystolicBP	DiastolicBP	FastingBloodSugar	ProteinInUrine	CholesterolTotal	Hemoglob
33	122	94	138.521516	1.562304	253.994490	1
171	129	112	126.524417	1.474631	199.434710	1
255	158	116	173.289446	0.968971	228.735784	1
282	148	84	172.205815	4.480833	221.115139	1
297	115	88	152.484900	2.311932	152.660029	1
299	143	104	129.129424	0.135343	171.618092	1
312	127	79	151.030044	2.491876	293.387461	1
313	167	118	176.954539	0.356023	281.175810	1
353	113	94	170.977158	1.578519	247.364350	1
552	174	116	164.694769	1.279343	174.134303	1
764	117	85	147.854874	1.166281	192.190323	1
811	172	79	196.589249	0.888195	168.759423	1
1111	140	112	164.756009	3.153684	182.097982	1
1270	154	81	150.174373	4.983143	204.190158	1
1271	143	61	147.802307	1.745105	155.428395	1
1296	156	83	154.022828	2.040151	207.221369	1
1335	127	94	133.011359	2.543581	220.677651	1
1604	112	80	140.335089	2.514646	262.899867	1

In [37]:
```
for j in E.columns:
    sns.histplot(E[j],bins=10,kde=True,edgecolor="black")
    plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in
a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [38]:
```
A['MedicalCheckupsFrequency'].max()
```

Out[38]:
```
3.9994688020115
```

In [36]:
```
A['MedicationAdherence'].max()
```

Out[36]:
```
9.99234480503182
```

In [37]:
```
f=A.loc[(A['DietQuality']>=7)&(A['SleepQuality']>=7)&(A['PhysicalActivi
f
```

Out[37]:

```
      PatientID  Age  Gender  Ethnicity  SocioeconomicStatus  EducationLevel        BMI  S
   5          6   22       0          0                    0               1  39.155643
  24         25   81       1          3                    0               0  17.998046
  79         80   82       1          3                    1               1  25.563485
 116        117   55       1          0                    0               3  17.238005
 167        168   89       0          3                    2               1  36.138780
 224        225   85       0          2                    2               2  33.709021
 227        228   80       0          0                    1               1  33.028982
 234        235   67       1          0                    2               1  36.149131
 312        313   34       1          0                    0               1  36.191008
 391        392   35       1          0                    1               0  27.481073
 394        395   51       0          1                    2               0  30.520470
 412        413   32       0          1                    2               2  23.505406
 421        422   87       0          1                    1               1  16.383745
 464        465   72       0          1                    1               2  30.066749
 536        537   50       1          2                    0               2  25.130378
 589        590   22       1          1                    0               1  27.814598
 631        632   24       0          0                    0               0  22.852745
 697        698   42       1          0                    0               1  32.171662
 716        717   27       1          1                    1               1  16.412086
 721        722   71       0          0                    0               2  25.038543
 732        733   89       0          0                    2               2  20.339183
 827        828   34       1          0                    2               3  39.733516
 849        850   76       0          0                    1               3  15.905182
 867        868   48       0          0                    2               3  27.002254
 908        909   20       0          1                    1               1  38.272454
 981        982   51       1          0                    1               2  31.230719
 983        984   53       0          3                    1               2  15.841243
1004       1005   73       0          0                    0               2  32.456815
1065       1066   64       0          0                    2               1  28.195184
1073       1074   84       1          1                    0               1  39.196559
1082       1083   30       1          2                    2               2  22.344047
1259       1260   78       1          0                    0               2  16.647350
1265       1266   56       0          1                    1               2  21.424793
1277       1278   59       1          0                    1               0  26.099705
1283       1284   78       1          0                    2               2  23.220045
1295       1296   24       1          1                    0               3  28.447988
1307       1308   21       1          0                    0               2  27.427233
1310       1311   85       0          1                    1               1  36.984925
1313       1314   46       0          0                    0               2  16.987993
```

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
1363	1364	66	1	0	2	2	37.424852	
1437	1438	39	1	0	0	0	18.824047	
1439	1440	59	1	1	2	1	27.756177	
1448	1449	57	0	3	2	2	22.053890	
1451	1452	63	0	1	2	2	15.769379	
1477	1478	44	1	1	2	1	36.160879	
1540	1541	79	0	0	1	1	34.499012	

In [38]:
```python
f['Smoking'].value_counts()
```

Out[38]:
```
Smoking
0    31
1    15
Name: count, dtype: int64
```

In [39]:
```python
f.loc[f['QualityOfLifeScore']<=20]
```

Out[39]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
5	6	22	0	0	0	1	39.155643	
394	395	51	0	1	2	0	30.520470	
908	909	20	0	1	1	1	38.272454	
981	982	51	1	0	1	2	31.230719	
1004	1005	73	0	0	0	2	32.456815	
1448	1449	57	0	3	2	2	22.053890	
1540	1541	79	0	0	1	1	34.499012	

In [40]:
```
f1=A.loc[(A['DietQuality']<=7)&(A['SleepQuality']<=7)&(A['PhysicalActiv
f1
```

Out[40]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	Si
0	1	71	0	0	0	2	31.069414	
3	4	40	0	2	0	1	31.329680	
4	5	43	0	1	1	2	23.726311	
7	8	72	1	0	1	3	30.760440	
11	12	21	1	2	0	3	16.799520	
...	...	...	...	...	...	...	...	
1646	1647	87	0	1	1	0	33.894781	
1647	1648	23	0	3	0	1	39.109585	
1652	1653	20	0	0	1	3	20.378015	
1653	1654	73	1	0	1	3	35.634449	
1657	1658	90	0	0	2	2	24.964149	

494 rows × 53 columns

In [41]:
```
f1['Diagnosis'].value_counts()
```

Out[41]:
```
Diagnosis
1    456
0     38
Name: count, dtype: int64
```

In [42]:
```
FQ=f1.loc[f1['QualityOfLifeScore']>=60]
FQ
```

Out[42]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	Si
0	1	71	0	0	0	2	31.069414	
3	4	40	0	2	0	1	31.329680	
7	8	72	1	0	1	3	30.760440	
11	12	21	1	2	0	3	16.799520	
18	19	68	1	1	2	2	34.267299	
...	...	...	...	...	...	...	...	
1632	1633	33	0	0	2	0	15.148205	
1641	1642	52	0	3	1	2	18.688796	
1646	1647	87	0	1	1	0	33.894781	
1652	1653	20	0	0	1	3	20.378015	
1653	1654	73	1	0	1	3	35.634449	

195 rows × 53 columns

In [43]:
```
FQ['Diagnosis'].value_counts()
```

Out[43]:
```
Diagnosis
1    178
0     17
Name: count, dtype: int64
```

In [44]:
```
FQ1=f1.loc[f1['QualityOfLifeScore']<=60]
FQ1
```

Out[44]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
4	5	43	0	1	1	2	23.726311	
14	15	40	0	2	0	3	27.000463	
36	37	23	1	0	1	3	29.403983	
45	46	27	0	0	1	1	39.014213	
52	53	25	0	0	1	1	18.073047	
...	...	...	...	...	...	...	...	
1633	1634	72	1	0	2	2	29.811019	
1635	1636	88	0	3	2	3	23.922497	
1637	1638	85	1	0	1	1	25.521931	
1647	1648	23	0	3	0	1	39.109585	
1657	1658	90	0	0	2	2	24.964149	

299 rows × 53 columns

In [45]:
```
FQ['Diagnosis'].value_counts()
```

Out[45]:
```
Diagnosis
1    178
0     17
Name: count, dtype: int64
```

In [46]:
```

T=A.loc[(A['Age']>60)&(A['BMI']>=30)&(A['QualityOfLifeScore']>=80)&(A[
T
```

Out[46]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI
2	3	80	1	1	0	1	37.394822
31	32	70	0	2	1	0	35.335172
141	142	81	0	2	2	1	36.073750
190	191	71	0	0	0	1	32.911325
329	330	73	0	0	0	0	33.237923
355	356	77	1	1	1	1	32.136013
532	533	66	1	0	1	1	38.239577
550	551	79	0	0	1	1	32.005944
625	626	70	0	3	2	3	31.089406
693	694	78	1	0	2	2	35.974277
718	719	61	0	0	0	1	36.075924
761	762	87	1	0	0	2	34.345019
834	835	70	0	2	0	3	35.972197
851	852	74	1	0	2	1	32.717021
869	870	87	1	0	2	0	33.049424
911	912	74	1	1	0	2	34.275959
956	957	70	1	1	0	1	34.771242
1020	1021	88	1	0	0	2	38.644616
1108	1109	63	0	0	0	2	36.249690
1135	1136	90	0	0	1	1	36.787680
1266	1267	79	1	0	1	2	31.603819
1323	1324	79	1	3	1	1	33.975701
1404	1405	79	0	3	0	3	37.837774
1419	1420	83	1	0	1	1	35.823291
1481	1482	64	0	2	0	1	32.619890
1572	1573	66	1	0	0	2	35.817081
1654	1655	90	0	0	1	2	39.677059

In [47]:
```
T['Gender'].value_counts()
```

Out[47]:
```
Gender
1    14
0    13
Name: count, dtype: int64
```

In [48]:
```
T.shape
```

Out[48]:
```
(27, 53)
```

In [49]:
```
t=A.loc[(A['Age']<60)&(A['BMI']>=30)&(A['QualityOfLifeScore']>=80)&(A[
t
```

Out[49]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
3	4	40	0	2	0	1	31.329680	
89	90	31	1	0	1	2	36.033493	
101	102	20	1	0	1	0	33.473207	
173	174	57	0	0	0	2	35.860728	
193	194	58	0	0	0	1	35.328750	
212	213	38	0	3	0	1	32.036503	
220	221	55	1	0	1	1	34.241054	
232	233	33	0	0	2	1	33.084867	
289	290	21	1	1	1	3	31.899812	
371	372	48	1	0	0	2	31.780539	
379	380	52	0	0	0	1	39.357624	
390	391	23	1	0	1	2	33.490542	
399	400	52	0	1	2	0	37.061755	
438	439	32	1	0	1	1	38.989855	
470	471	54	1	0	1	1	36.667284	
474	475	36	1	3	0	1	36.010008	
509	510	58	0	0	0	1	31.763513	
548	549	27	1	0	1	1	33.249831	
616	617	42	0	0	2	3	35.141884	
637	638	36	0	2	0	2	33.204584	
760	761	45	1	0	1	2	31.312118	
764	765	31	0	3	2	3	31.069342	
843	844	48	1	1	2	2	30.393151	
873	874	55	1	2	2	1	35.142620	
934	935	43	0	3	1	1	34.036471	
949	950	39	0	1	1	2	33.260344	
998	999	54	0	2	1	3	34.613413	
1051	1052	57	0	2	1	2	36.995261	
1096	1097	45	0	2	0	0	38.906149	
1145	1146	58	1	1	0	1	31.731427	
1174	1175	37	1	3	1	2	30.283925	
1175	1176	51	1	1	2	1	39.101353	
1198	1199	44	1	2	0	1	37.465414	
1249	1250	48	0	0	2	3	36.483320	
1298	1299	50	1	2	2	1	33.420128	
1485	1486	51	1	0	1	1	34.289722	
1560	1561	33	0	0	1	3	38.006809	
1577	1578	47	0	0	2	3	32.908862	
1580	1581	39	0	1	1	2	37.920164	

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	Sı
1587	1588	54	1	0	2	2	38.330627	
1628	1629	52	0	2	0	2	35.979809	

In [50]:
```
t['Gender'].value_counts()
```

Out[50]:
```
Gender
0    21
1    20
Name: count, dtype: int64
```

In [51]:
```
t.shape
```

Out[51]:
```
(41, 53)
```

In [52]:
```
e=A.loc[(A['Age']<60)&(A['BMI']>=30)&(A['QualityOfLifeScore']<=60)&(A[
e
```

Out[52]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	Sı
5	6	22	0	0	0	1	39.155643	
6	7	41	0	1	0	1	35.040487	
10	11	57	1	1	0	3	31.749248	
41	42	21	1	2	1	3	39.819275	
48	49	55	1	0	0	2	35.742392	
...	...	...	...	...	...	...	...	
1571	1572	56	0	0	1	2	33.090236	
1604	1605	42	1	0	1	2	38.555843	
1612	1613	20	0	0	2	2	36.796656	
1631	1632	50	1	0	2	0	33.037229	
1644	1645	32	0	0	1	3	32.164663	

139 rows × 53 columns

In [53]:
```
e['Smoking'].value_counts()
```

Out[53]:
```
Smoking
0    106
1     33
Name: count, dtype: int64
```

In [54]:
```
e['FamilyHistoryKidneyDisease'].value_counts()
```

Out[54]:
```
FamilyHistoryKidneyDisease
0    117
1     22
Name: count, dtype: int64
```

In [55]:
```
e.shape
```

Out[55]:
```
(139, 53)
```

In [56]:
```
w=A.loc[(A['Age']>60)&(A['BMI']>=30)&(A['QualityOfLifeScore']<=60)&(A[
w
```

Out[56]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
23	24	81	0	0	1	2	30.170664	
26	27	81	1	0	0	1	36.691040	
60	61	81	1	2	0	2	39.622869	
73	74	64	1	0	1	3	36.391092	
96	97	63	0	0	2	1	33.045011	
...	...	...	...	...	...	...	...	
1499	1500	82	1	1	2	2	30.688901	
1540	1541	79	0	0	1	1	34.499012	
1568	1569	76	1	0	2	0	35.889120	
1589	1590	78	0	0	1	3	37.853912	
1599	1600	76	0	0	1	3	36.219781	

95 rows × 53 columns

In [57]:
```
w.shape
```

Out[57]:
```
(95, 53)
```

In [58]:
```
A['AlcoholConsumption'].max()

```

Out[58]:
```
19.99271250850888
```

In [59]:
```
b=A.loc[(A['Age']>60)&(A['BMI'].between(19,30))&(A['QualityOfLifeScore
b
```

Out[59]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
138	139	89	1	0	2	1	19.585134	
246	247	70	1	0	0	0	22.310377	
287	288	85	1	0	0	1	24.277652	
627	628	71	1	0	2	2	25.906111	
756	757	64	1	0	2	2	25.976037	
878	879	88	0	0	2	0	19.972293	
1084	1085	78	0	3	0	2	27.505787	
1102	1103	62	1	0	1	1	29.441536	
1288	1289	72	0	0	2	3	22.817067	
1400	1401	83	0	0	1	3	22.328899	

In [60]:
```
b.shape
```

Out[60]:
```
(10, 53)
```

In [61]:
```
A.loc[(A['Age']<60)&(A['BMI'].between(19,30))&(A['QualityOfLifeScore']
```

Out[61]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
17	18	41	1	0	1	3	26.046226	
201	202	26	1	3	1	3	20.318174	
216	217	43	1	0	1	3	19.648341	
235	236	39	0	1	2	1	25.687643	
258	259	38	0	0	1	2	27.514522	
269	270	43	1	0	1	1	23.739630	
303	304	26	1	0	0	3	24.357650	
391	392	35	1	0	1	0	27.481073	
490	491	36	1	0	2	3	22.170394	
959	960	51	1	2	0	2	20.105966	
1110	1111	31	0	3	2	2	23.909166	
1260	1261	57	1	0	2	2	19.176392	
1317	1318	54	1	0	0	3	23.232633	
1347	1348	52	0	0	2	2	24.466060	
1354	1355	48	0	0	0	1	29.714652	
1651	1652	42	1	1	1	3	21.653960	

In [62]:
```
A.loc[(A['Age']>60)&(A['BMI']<19)&(A['QualityOfLifeScore']>=80)&(A['Fas
```

Out[62]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
94	95	81	1	0	0	0	15.605151	
144	145	63	1	1	2	3	15.136093	
615	616	66	1	0	1	2	17.189389	
677	678	75	0	0	0	2	17.109699	
1114	1115	68	1	0	2	3	16.661828	
1272	1273	81	0	3	1	1	16.979660	
1472	1473	79	1	0	0	2	16.752951	

In [63]:
```
U=A.loc[A['Diagnosis']==0]
U
```

Out[63]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
7	8	72	1	0	1	3	30.760440	
14	15	40	0	2	0	3	27.000463	
35	36	37	0	0	1	2	17.446425	
49	50	69	1	3	1	0	23.745084	
66	67	72	0	1	2	3	20.854976	
...	...	...	...	...	...	...	...	
1649	1650	89	0	0	1	2	22.859626	
1651	1652	42	1	1	1	3	21.653960	
1654	1655	90	0	0	1	2	39.677059	
1655	1656	34	0	0	2	1	28.922015	
1656	1657	84	0	0	2	3	21.951219	

135 rows × 53 columns

In [64]:
```
AS=A.loc[(A['SerumCreatinine']>=1.1)|(A['BUNLevels']>=20)&(A['GFR']<90)
AS
```

Out[64]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
0	1	71	0	0	0	2	31.069414	
1	2	34	0	0	1	3	29.692119	
2	3	80	1	1	0	1	37.394822	
3	4	40	0	2	0	1	31.329680	
4	5	43	0	1	1	2	23.726311	
...	...	...	...	...	...	...	...	
1653	1654	73	1	0	1	3	35.634449	
1654	1655	90	0	0	1	2	39.677059	
1656	1657	84	0	0	2	3	21.951219	
1657	1658	90	0	0	2	2	24.964149	
1658	1659	34	1	1	0	0	19.253258	

1529 rows × 53 columns

In [65]:
```
AS['Diagnosis'].value_counts()
```

Out[65]:
```
Diagnosis
1    1441
0      88
Name: count, dtype: int64
```

In [66]:
```
As1=A.loc[(A['SerumCreatinine']<=1.1)|(A['BUNLevels']<=20)&(A['GFR']>9
As1
```

Out[66]:

	PatientID	Age	Gender	Ethnicity	SocioeconomicStatus	EducationLevel	BMI	S
14	15	40	0	2	0	3	27.000463	
17	18	41	1	0	1	3	26.046226	
27	28	70	1	1	2	1	20.814202	
28	29	74	1	0	1	2	30.192398	
30	31	22	0	1	1	2	20.196093	
...	...	...	...	...	...	...	...	
1646	1647	87	0	1	1	0	33.894781	
1649	1650	89	0	0	1	2	22.859626	
1651	1652	42	1	1	1	3	21.653960	
1655	1656	34	0	0	2	1	28.922015	
1656	1657	84	0	0	2	3	21.951219	

236 rows × 53 columns

In [67]:
```
As1['Diagnosis'].value_counts()
```

Out[67]:
```
Diagnosis
1    181
0     55
Name: count, dtype: int64
```

In [68]:
```
A.drop(columns=['Ethnicity','SocioeconomicStatus','EducationLevel'],inp
```

In [69]:
```
A
```

Out[69]:

	PatientID	Age	Gender	BMI	Smoking	AlcoholConsumption	PhysicalActivity	Di
0	1	71	0	31.069414	1	5.128112	1.676220	
1	2	34	0	29.692119	1	18.609552	8.377574	
2	3	80	1	37.394822	1	11.882429	9.607401	
3	4	40	0	31.329680	0	16.020165	0.408871	
4	5	43	0	23.726311	0	7.944146	0.780319	
...	...	...	...	...	...	...	...	
1654	1655	90	0	39.677059	1	1.370151	4.157954	
1655	1656	34	0	28.922015	0	3.372073	9.647525	
1656	1657	84	0	21.951219	0	15.825955	7.349964	
1657	1658	90	0	24.964149	0	12.967462	0.618614	
1658	1659	34	1	19.253258	1	11.396510	7.446314	

1659 rows × 50 columns

In [70]:
```
A.corr().head()
```

Out[70]:

	PatientID	Age	Gender	BMI	Smoking	AlcoholConsumption	Physica
PatientID	1.000000	0.001166	-0.023822	-0.036264	-0.005863	0.029087	-
Age	0.001166	1.000000	-0.037765	-0.033202	0.020215	-0.006030	
Gender	-0.023822	-0.037765	1.000000	-0.023381	0.004054	-0.020473	
BMI	-0.036264	-0.033202	-0.023381	1.000000	-0.000968	-0.067239	
Smoking	-0.005863	0.020215	0.004054	-0.000968	1.000000	0.032707	

In [71]:
```
Fs=A.drop(columns='Edema',axis=1)
T=A['Edema']
```

In [72]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(Fs,T,train_size=0.90,rar
```

In [73]:
```
from sklearn.preprocessing import MinMaxScaler
M=MinMaxScaler()
```

In [74]:
```
X_train[["FastingBloodSugar","PatientID","ACR","SystolicBP","Diastolic
X_train
```

Out[74]:

	PatientID	Age	Gender	BMI	Smoking	AlcoholConsumption	PhysicalActivity	Di
929	0.560048	36	0	21.230094	0	9.040571	3.563935	
562	0.338564	54	0	29.274789	0	11.233628	6.253564	
1156	0.697043	62	0	21.961632	0	17.860599	1.428099	
234	0.140616	67	1	36.149131	0	2.646611	9.370097	
155	0.092939	22	1	35.839353	0	2.351992	4.423244	
...	...	...	...	...	...	...	...	
923	0.556427	82	0	32.216126	0	4.487337	8.215404	
1081	0.651780	59	0	26.720474	1	17.295340	9.340598	
449	0.270368	38	0	35.146443	1	5.031979	8.372899	
396	0.238383	31	1	29.238032	0	13.517838	8.622992	
1478	0.891370	64	0	27.261161	1	14.909048	1.307203	

1493 rows × 49 columns

In [75]:
```
X_test[["FastingBloodSugar","PatientID","ACR","SystolicBP","DiastolicB
X_test
```

Out[75]:

	PatientID	Age	Gender	BMI	Smoking	AlcoholConsumption	PhysicalActivity	Di
743	0.447797	55	1	21.726306	1	0.377445	0.361527	
1487	0.896801	63	0	35.299537	1	0.995805	0.958054	
1082	0.652384	30	1	22.344047	1	17.702386	8.542203	
1485	0.895594	51	1	34.289722	0	6.797916	9.548392	
30	0.017502	22	0	20.196093	0	9.701107	3.116833	
...	...	...	...	...	...	...	...	
263	0.158117	36	0	28.564386	0	14.791730	4.036793	
48	0.028365	55	1	35.742392	0	16.650567	9.069743	
1317	0.794206	54	1	23.232633	1	11.872264	1.596429	
86	0.051298	60	1	32.096919	0	14.891137	4.832487	
1398	0.843090	90	1	34.767573	0	2.305825	7.446408	

166 rows × 49 columns

In [76]:
```
from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import LogisticRegression

Log=LogisticRegression()
params={'C':[0.2,0.006,0.001,0.4],"penalty":["l1","l2"]}

G=GridSearchCV(Log,param_grid=params,scoring="accuracy",cv=7)
```

In [77]:
```
G.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logist
ic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression (https://scikit-learn.org/stable/modules/linear_model.html#l
ogistic-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logist
ic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
```

In [78]:
```
G.best_params_
```

Out[78]:
```
{'C': 0.006, 'penalty': 'l2'}
```

In [79]:
```
H=G.best_estimator_
H
```

Out[79]:
```
LogisticRegression(C=0.006)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [80]:
```
p=H.predict(X_test)
p
```

Out[80]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [81]:
```
H.score(X_train,y_train)
```

Out[81]:
```
0.797052913596785
```

In [82]:
```
H.score(X_test,y_test)
```

Out[82]:
```
0.8072289156626506
```

In [83]:
```
from sklearn.metrics import accuracy_score,classification_report,confus
```

In [84]:
```
accuracy_score(y_test,p)
```

Out[84]:
```
0.8072289156626506
```

In [85]:
```
print(classification_report(y_test,p))
```

```
              precision    recall  f1-score   support

           0       0.81      1.00      0.89       134
           1       0.00      0.00      0.00        32

    accuracy                           0.81       166
   macro avg       0.40      0.50      0.45       166
weighted avg       0.65      0.81      0.72       166


C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [86]:
```
confusion_matrix(y_test,p)
```

Out[86]:
```
array([[134,   0],
       [ 32,   0]], dtype=int64)
```

In [87]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
K=KNeighborsClassifier()
params={'n_neighbors':[3,5,7,11]}
G=GridSearchCV(K,param_grid=params,scoring="accuracy",cv=6)

```

In [88]:
```
G.fit(X_train,y_train)
```

Out[88]:
```
GridSearchCV(cv=6, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': [3, 5, 7, 11]}, scoring='accurac
y')
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [89]:
```
G.best_params_
```

Out[89]:
```
{'n_neighbors': 11}
```

In [90]:
```
Model=G.best_estimator_
Model
```

Out[90]:
```
KNeighborsClassifier(n_neighbors=11)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [91]:
```
preds=Model.predict(X_test)
preds
```

Out[91]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [92]:
```
Model.score(X_train,y_train)
```

Out[92]:
```
0.7937039517749498
```

In [93]:
```
Model.score(X_test,y_test)
```

Out[93]:
```
0.8012048192771084
```

In [94]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
S=SVC()
params={"gamma":[0.1,0.3,0.4,0.5],"kernel":["rbf"]}

G=GridSearchCV(S,param_grid=params,scoring="accuracy",cv=6)
```

In [95]:
```
G.fit(X_train,y_train)
```

Out[95]:
```
GridSearchCV(cv=6, estimator=SVC(),
             param_grid={'gamma': [0.1, 0.3, 0.4, 0.5], 'kernel': ['rb
f']},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [96]:
```
G.best_params_
```

Out[96]:
```
{'gamma': 0.1, 'kernel': 'rbf'}
```

In [97]:
```
M=G.best_estimator_
M
```

Out[97]:
```
SVC(gamma=0.1)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [98]:
```
pred=M.predict(X_test)
pred
```

Out[98]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [99]:
```
M.score(X_train,y_train)
```

Out[99]:
```
1.0
```

In [100]:
```
M.score(X_test,y_test)
```

Out[100]:
```
0.8072289156626506
```

In [101]:
```
from sklearn.naive_bayes import GaussianNB
A= GaussianNB()
```

In [102]:
```
A.fit(X_train,y_train)
```

Out[102]:
```
GaussianNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [103]:
```
A.score(X_train,y_train)
```

Out[103]:
```
0.7843268586738111
```

In [104]:
```
A.score(X_test,y_test)
```

Out[104]:
```
0.7831325301204819
```

In [105]:
```
from sklearn.naive_bayes import BernoulliNB
B=BernoulliNB()
```

In [106]:
```
B.fit(X_train,y_train)
```

Out[106]:
```
BernoulliNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [107]:
```
B.score(X_train,y_train)
```

Out[107]:
```
0.797722705961152
```

In [108]:
```
B.score(X_test,y_test)
```

Out[108]:
```
0.8072289156626506
```

In [109]:
```
from sklearn.naive_bayes import CategoricalNB
C=CategoricalNB()
```

In [110]:
```
C.fit(X_train,y_train)
```

Out[110]:
```
CategoricalNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [111]:
```
C.score(X_train,y_train)
```

Out[111]:
```
0.8225050234427328
```

In [112]:
```
C.score(X_test,y_test)
```

Out[112]:
```
0.8012048192771084
```

In [113]:
```
from sklearn.naive_bayes import ComplementNB
c=ComplementNB()
```

In [114]:
```
c.fit(X_train,y_train)
```

Out[114]:
```
ComplementNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [115]:
```
c.score(X_train,y_train)
```

Out[115]:
```
0.5331547220361688
```

In [116]:
```
c.score(X_test,y_test)
```

Out[116]:
```
0.5240963855421686
```

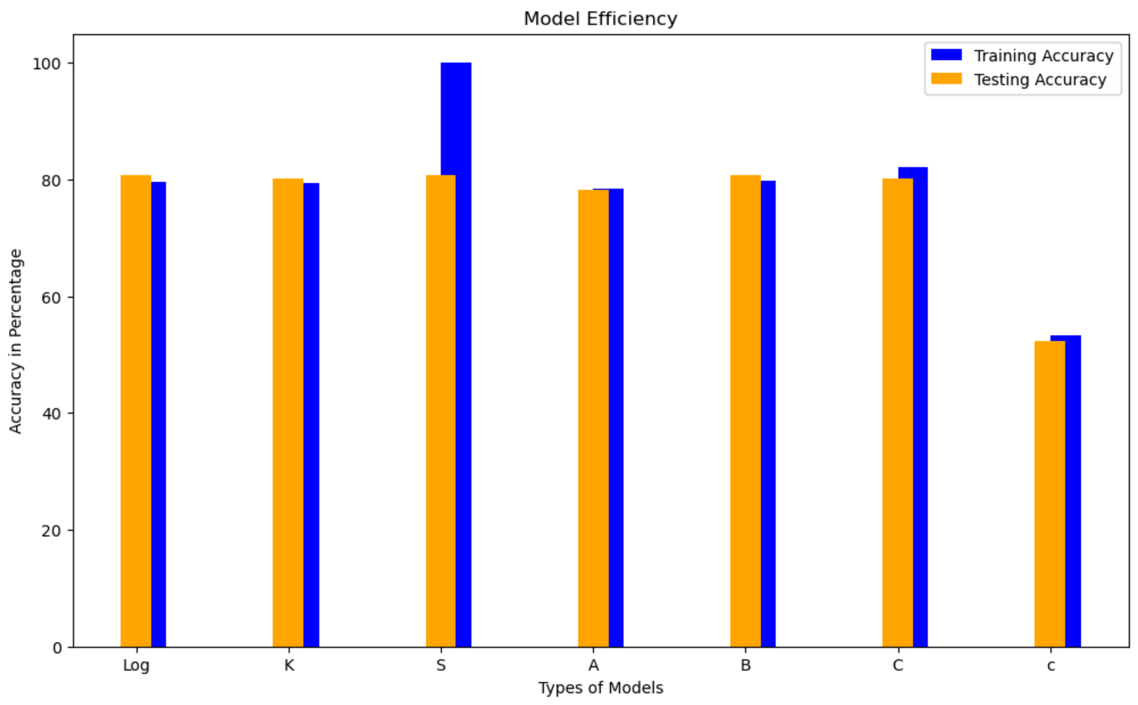
In [117]:
```
Z={'models':["Log","K","S","A","B","C","c"],"Train":[79.70,79.37,100,78
```

In [118]:
```
Z=pd.DataFrame(Z)
Z
```

Out[118]:

	models	Train	Test
0	Log	79.70	80.72
1	K	79.37	80.12
2	S	100.00	80.72
3	A	78.43	78.31
4	B	79.77	80.72
5	C	82.25	80.12
6	c	53.31	52.40

In [119]:
```
plt.figure(figsize=(12,7))
plt.bar(Z['models'],Z['Train'],color="blue",width=0.2,align="edge",labe
plt.bar(Z['models'],Z['Test'],color="orange",width=0.2,align="center",l
plt.legend()
plt.title("Model Efficiency")
plt.xlabel('Types of Models')
plt.ylabel('Accuracy in Percentage')
plt.show()
```



In [120]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
D=DecisionTreeClassifier()
params={'max_depth':[5,7,10,12],'criterion':['gini'],'min_samples_split
G=GridSearchCV(D,param_grid=params,scoring='accuracy',cv=6)

```

In [121]:
```
G.fit(X_train,y_train)
```

Out[121]:
```
GridSearchCV(cv=6, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini'], 'max_depth': [5, 7, 10, 1
2],
                         'min_samples_split': [2, 5, 8, 9]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [122]:
```
G.best_params_
```

Out[122]:
```
{'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 2}
```

In [123]:
```
l=G.best_estimator_
l
```

Out[123]:
```
DecisionTreeClassifier(max_depth=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [124]:
```
pred=l.predict(X_test)
pred
```

Out[124]:
```
array([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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [125]:
```
l.score(X_train,y_train)
```

Out[125]:
```
0.8191560616208975
```

In [126]:
```
l.score(X_test,y_test)
```

Out[126]:
```
0.8132530120481928
```

In [127]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
D=DecisionTreeClassifier()
params={'max_depth':[7,10,12],'criterion':['entropy'],'min_samples_spli
g=GridSearchCV(D,param_grid=params,scoring='accuracy',cv=6)

```

In [128]:
```
g.fit(X_train,y_train)
```

Out[128]:
```
GridSearchCV(cv=6, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['entropy'], 'max_depth': [7, 10, 1
2],
                         'min_samples_split': [2, 8, 9]},
             scoring='accuracy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [129]:
```
g.best_params_
```

Out[129]:
```
{'criterion': 'entropy', 'max_depth': 7, 'min_samples_split': 9}
```

In [130]:
```
f=g.best_estimator_
f
```

Out[130]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=7, min_samples_split
=9)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [131]:
```
predict=f.predict(X_test)
predict
```

Out[131]:
```
array([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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [132]:
```
f.score(X_train,y_train)
```

Out[132]:
```
0.8238446081714669
```

In [133]:
```
f.score(X_test,y_test)
```

Out[133]:
```
0.8132530120481928
```

In [134]:
```
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
A= RandomForestClassifier(n_estimators=25)
```

In [135]:
```
A.fit(X_train,y_train)
```

Out[135]:
```
RandomForestClassifier(n_estimators=25)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [136]:
```
A.score(X_train,y_train)
```

Out[136]:
```
0.9986604152712659
```

In [137]:
```
A.score(X_test,y_test)
```

Out[137]:
```
0.7951807228915663
```

In [138]:
```
from sklearn.ensemble import RandomForestClassifier
A1= RandomForestClassifier(n_estimators=80)
```

In [139]:
```
A1.fit(X_train,y_train)
```

Out[139]:
```
RandomForestClassifier(n_estimators=80)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [140]:
```
A1.score(X_train,y_train)
```

Out[140]:
```
1.0
```

In [141]:
```
A1.score(X_test,y_test)
```

Out[141]:
```
0.8072289156626506
```

In [142]:
```
B=BaggingClassifier(estimator=Log,n_estimators=30)
B.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logist
ic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression (https://scikit-learn.org/stable/modules/linear_model.html#l
ogistic-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logist
ic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
```

In [143]:
```
B.score(X_train,y_train)
```

Out[143]:
```
0.796383121232418
```

In [144]:
```
B.score(X_test,y_test)
```

Out[144]:
```
0.8072289156626506
```

In [145]:
```
from sklearn.ensemble import VotingClassifier
Z=VotingClassifier(estimators=[('log',LogisticRegression()),('KNN',KNe
```

In [146]:
```
Z.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[146]:
```
VotingClassifier(estimators=[('log', LogisticRegression()),
                             ('KNN', KNeighborsClassifier()),
                             ('NB', GaussianNB())])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [147]:
```
Z.score(X_train,y_train)
```

Out[147]:
```
0.8010716677829873
```

In [148]:
```
Z.score(X_test,y_test)
```

Out[148]:
```
0.7951807228915663
```

In [149]:
```
from sklearn.ensemble import StackingClassifier
y=StackingClassifier(estimators=[('NB',GaussianNB()),('Svc',SVC()),('KN
```

In [150]:
```
y.fit(X_train,y_train)
```

Out[150]:
```
StackingClassifier(estimators=[('NB', GaussianNB()), ('Svc', SVC()),
                               ('KNN', KNeighborsClassifier())],
                   final_estimator=LogisticRegression())
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [151]:
```
y.score(X_train,y_train)
```

Out[151]:
```
0.797052913596785
```

In [152]:
```
y.score(X_test,y_test)
```

Out[152]:
```
0.8072289156626506
```

In [153]:
```
b=BaggingClassifier(estimator=S,n_estimators=75)
b.fit(X_train,y_train)
```

Out[153]:
```
BaggingClassifier(estimator=SVC(), n_estimators=75)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [154]:
```
b.score(X_train,y_train)
```

Out[154]:
```
0.797052913596785
```

In [155]:
```
b.score(X_test,y_test)
```

Out[155]:
```
0.8072289156626506
```

In [156]:
```
from sklearn.ensemble import VotingClassifier
R=VotingClassifier(estimators=[('KNN',KNeighborsClassifier()),('S',SVC(
```

In [157]:
```
R.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[157]:
```
VotingClassifier(estimators=[('KNN', KNeighborsClassifier()), ('S', SVC
()),
                             ('NB', GaussianNB()),
                             ('log', LogisticRegression())])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [158]:
```
R.score(X_train,y_train)
```

Out[158]:
```
0.797052913596785
```

In [159]:
```
R.score(X_test,y_test)
```

Out[159]:
```
0.8072289156626506
```

In [160]:
```
from sklearn.naive_bayes import MultinomialNB
M=MultinomialNB()
```

In [161]:
```
M.fit(X_train,y_train)
```

Out[161]:
```
MultinomialNB()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [162]:
```
from sklearn.ensemble import StackingClassifier
p=StackingClassifier(estimators=[('log',LogisticRegression()),('NB',Ga
```

In [163]:
```
p.fit(X_train,y_train)

```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[163]:

```
StackingClassifier(estimators=[('log', LogisticRegression()),
                               ('NB', GaussianNB()), ('Svc', SVC()),
                               ('KNN', KNeighborsClassifier())],
                   final_estimator=SVC())
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [164]:

```
p.score(X_train,y_train)
```

Out[164]:

```
0.797052913596785
```

In [165]:

```
p.score(X_test,y_test)
```

Out[165]:

```
0.8072289156626506
```

In [166]:

```
from sklearn.ensemble import StackingClassifier
I=StackingClassifier(estimators=[('NB',ComplementNB()),('B',BernoulliNE
```

In [167]:

```
I.fit(X_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[167]:
```
StackingClassifier(estimators=[('NB', ComplementNB()), ('B', BernoulliNB
()),
                               ('m', MultinomialNB()), ('Svc', SVC()),
                               ('KNN', KNeighborsClassifier()),
                               ('l', LogisticRegression())],
                   final_estimator=LogisticRegression())
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [168]:
```python
I.score(X_train,y_train)
```

Out[168]:
```
0.797052913596785
```

In [169]:
```python
I.score(X_test,y_test)
```

Out[169]:
```
0.8072289156626506
```

In [170]:
```python
from sklearn.ensemble import AdaBoostClassifier
AC=AdaBoostClassifier(n_estimators=60)
AC.fit(X_train,y_train)
```

Out[170]:
```
AdaBoostClassifier(n_estimators=60)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [171]:
```python
AC.score(X_train,y_train)
```

Out[171]:
```
0.8111185532484929
```

In [172]:
```
AC.score(X_test,y_test)
```

Out[172]:
```
0.7771084337349398
```

```
# Conclusion_
#It's about chronic Kidney Disease data having all basic information about patients like age,ID,BP,Sugar,Cholesterol etc.

#APPROACH:-
          #firstly started to find effect of age,smoking,alcohol consumptionhabit on the health of individual but as studied further found different aspects affecting kidney,so started to sort whole data accordingly to get more insights.

#CONCLUSION:-
            #From analyzing whole data,I got to know that people in all Age group that are suffering from Obesity,BP,diabetes and Hypertension(stage 1 & 2) while majority of them having poor diet & sleep quality,lacking physical Activity So low quality of life score gives "ACR" results in range of 30-300,"GFR" shows majorly in (15,90),with High "Serum Creatinine" values like (>1.2) & "BUNlevels" blood urea Nitrogen levels are also high (>20),"CHolesterolTotal & cholesterol TRiglyceride"also according to age group is high, while smoking and alcohol consumption levels also affecting kidney.
            Thus from all these data,it is safe to say that 'ACR','GFR',"Serum Creatinine","BUNlevels","CHolesterolTotal & cholesterol TRiglyceride" and mostly neglecting medical checkups,lacking in adherence to proper medication,due to lack of health litracy,these are factors that indicating Kidney Diseases diagnosed in majority of patients, also high usage of medicine like NSAID and antibiotics etc also affect kidney.

```

In [ ]:
```

```