svc-knn-nb-dt-ensemble-learning-5

September 22, 2024

0.0.1 Why should I go for smart home devices?

Smart home devices gives convenience, energy savings, and good life with security. Making daily life more efficient and enjoyable

Target:- In this project discussion finding out

1. Less energy consumable devices
2. Their durability i.e. maximum life we call as device age
3. User preference with their efficiency
4. Conclusion from the above stu.

y. u ### Features (what we have):

1. Device type
2. Device usage hour per day
3. Energy consumption
4. Device age months
5. malfu

0.0.2 Approach inside:–

0.0.3 1. Less energy consumable devices =

a. From the given data, I am going to analyze device usages hour per day of all devices for finding less energy per day hour usage device, I found that device having less usage hour per day which is 19.88% from all devices2nd device is security system which consume 19.94% 3rd is light consume 20.00%, 4th is thermostat consume 20.09%, 5th is camera consume 20.

(So from all data I have found smart speaker 1108 entry data)

b. From the given data, I am going to analyze device energy consumption of all devices, I found that smart speaker device having less consumption which is 4.87% from all devices Then 2nd device is light which consume 5.04%, 3rd is camera consume 5.08%, 4th is security system consume 5.1%, 5th is thermostat consume 5. 14% (So from all data I have found smart speaker 1108 entry ### 2. Their durability i.e. maximum life we call as device age (which is depend on their malfunction incident)

c. From the given data, I am going to analyze device age months and malfunction incident of all devices,

I. I found that light device having less malfunction incident which is 2.02% but age months device is more less than other device, which is 29.91%

II. But when compare with smart speaker which is having next to light malfunction incident is 2.03 % more than light but age months is more than other devices which is 30.59%

0.0.4 3. User preference with their efficiency

a. From the given data, I am going to analyze user preferences and smart home efficiency of all devices,

I. I found that device thermostat having more preference, which is o.50 % by users and smart home efficient device 0.35 % but thermostat having more energy consumption having 5.14 % which is much more than all devices and per hour usages consumption which is 12.10% which is next to camer.

II. But in case of thermostat malfunction incident is more than all devices, which is 2.14% and device age months is 30.54% which is next to smart speaker

0.0.5 4. Conclusion

1. From the above analysis and discussion I conclude that smart speaker is a smart efficient device than all device. 2 As comparing with age and malfunction devise I found light having less defective device and device age is less but smart speaker having more ag e than the light so 1st option is smart speaker and next is light
2. Thermostat having more age device but more malfunction incident which is costly and more energy consumption. I thought thermostat is not bett er option to smart efficient device
3. Sequence of smart efficient devices is
4. Smart speaker
5. Light
6. Security system
7. Thermostat
8. Camera….

data) %

nction incidence

[41]:
```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
x=pd.read_csv(r"C:\Users\vidya\Downloads\smart_home_device_usage_data.csv")
x
```

[41]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	\
0	1	Smart Speaker	15.307188	1.961607	
1	2	Camera	19.973343	8.610689	
2	3	Security System	18.911535	2.651777	
3	4	Camera	7.011127	2.341653	
4	5	Camera	22.610684	4.859069	
…	…	…	…	…	

```
5398    5399       Thermostat          4.556314           5.871764
5399    5400           Lights          0.561856           1.555992
5400    5401    Smart Speaker         11.096236           7.677779
5401    5402  Security System          8.782169           7.467929
5402    5403       Thermostat         13.540381           9.043076

      UserPreferences  MalfunctionIncidents  DeviceAgeMonths  \
0                   1                     4               36
1                   1                     0               29
2                   1                     0               20
3                   0                     3               15
4                   1                     3               36
…                 …                     …              …
5398                1                     0               28
5399                1                     4               24
5400                0                     0               42
5401                0                     2               28
5402                0                     0               30

      SmartHomeEfficiency
0                       1
1                       1
2                       1
3                       0
4                       1
…                     …
5398                    0
5399                    0
5400                    0
5401                    1
5402                    0

[5403 rows x 8 columns]
```

[42]:
```
x.shape
```

[42]:
```
(5403, 8)
```

[45]:
```
x.columns
```

[45]:
```
Index(['UserID', 'DeviceType', 'UsageHoursPerDay', 'EnergyConsumption',
       'UserPreferences', 'MalfunctionIncidents', 'DeviceAgeMonths',
       'SmartHomeEfficiency'],
      dtype='object')
```

[77]:
```
x["DeviceType"].unique()
```

[77]:
```
array(['Smart Speaker', 'Camera', 'Security System', 'Thermostat',
       'Lights'], dtype=object)
```

[78]:
```
x.describe()
```

[78]:

	UserID	UsageHoursPerDay	EnergyConsumption	UserPreferences	\
count	5403.000000	5403.000000	5403.000000	5403.000000	
mean	2702.000000	12.052992	5.054302	0.511753	
std	1559.856083	6.714961	2.878941	0.499908	
min	1.000000	0.501241	0.101562	0.000000	
25%	1351.500000	6.297871	2.524968	0.000000	
50%	2702.000000	11.903768	5.007047	1.000000	
75%	4052.500000	17.791751	7.611912	1.000000	
max	5403.000000	23.987326	9.998071	1.000000	

	MalfunctionIncidents	DeviceAgeMonths	SmartHomeEfficiency
count	5403.000000	5403.000000	5403.000000
mean	2.066445	30.312234	0.376643
std	1.423291	16.990525	0.484589
min	0.000000	1.000000	0.000000
25%	1.000000	15.000000	0.000000
50%	2.000000	30.000000	0.000000
75%	3.000000	45.000000	1.000000
max	4.000000	59.000000	1.000000

[79]:
```
o=x.groupby(["DeviceType"])[["EnergyConsumption"]].mean().reset_index().
 ↪sort_values(by="EnergyConsumption",ascending=True)
o
```

[79]:

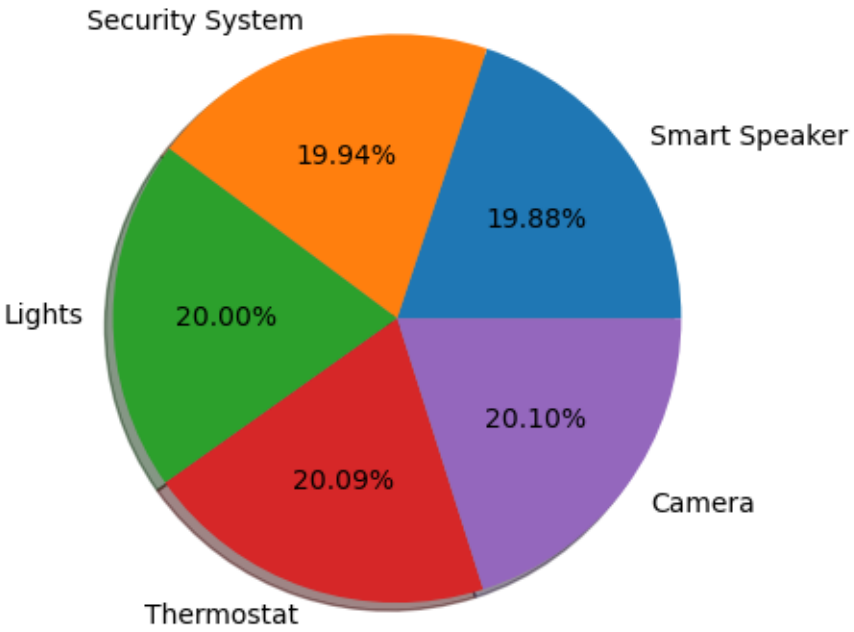
	DeviceType	EnergyConsumption
3	Smart Speaker	4.872036
1	Lights	5.044884
0	Camera	5.080666
2	Security System	5.138192
4	Thermostat	5.144356

[80]:
```
o=x.groupby(["DeviceType"])[["UsageHoursPerDay"]].mean().reset_index().
 ↪sort_values(by="UsageHoursPerDay",ascending=True)
o
```

[80]:

	DeviceType	UsageHoursPerDay
3	Smart Speaker	11.979308
2	Security System	12.016149
1	Lights	12.052646
4	Thermostat	12.105753
0	Camera	12.113435

[81]:
```
plt.pie(o["UsageHoursPerDay"],labels=o["DeviceType"],shadow=True,autopct="%.
↪2f%%")
plt.show()
```



1 Device confirm = Smart speaker is having less usage house per day

[83]:
```
i=x.loc[(x["DeviceType"]=="Smart Speaker")]
i
```

[83]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	\
0	1	Smart Speaker	15.307188	1.961607	
10	11	Smart Speaker	1.446710	7.723881	
14	15	Smart Speaker	22.494525	1.468928	
16	17	Smart Speaker	11.810032	8.228216	
19	20	Smart Speaker	1.018554	1.344045	
…	…	…	…	…	
5383	5384	Smart Speaker	23.229510	4.061440	
5389	5390	Smart Speaker	5.927129	0.364262	
5390	5391	Smart Speaker	17.972310	4.712130	
5395	5396	Smart Speaker	17.317435	8.839776	
5400	5401	Smart Speaker	11.096236	7.677779	

```
      UserPreferences  MalfunctionIncidents  DeviceAgeMonths  \
0                   1                     4               36
10                  0                     3               54
14                  0                     0               19
16                  1                     2                3
19                  1                     1                3
…                 …                     …              …
5383                0                     1               39
5389                0                     1               44
5390                0                     0               40
5395                1                     0               13
5400                0                     0               42

      SmartHomeEfficiency
0                       1
10                      0
14                      1
16                      1
19                      1
…                     …
5383                    1
5389                    0
5390                    0
5395                    0
5400                    0

[1108 rows x 8 columns]
```

[84]:
```
x.groupby(["DeviceType"])[["MalfunctionIncidents"]].mean().reset_index().
 ↪sort_values(by="MalfunctionIncidents",ascending=True)
```

[84]:
```
        DeviceType  MalfunctionIncidents
1           Lights              2.021159
3    Smart Speaker              2.037906
2  Security System              2.048689
0           Camera              2.087193
4       Thermostat              2.140520
```

[85]:
```
x.groupby(["DeviceType"])[["DeviceAgeMonths"]].mean().reset_index().
 ↪sort_values(by="DeviceAgeMonths",ascending=True)
```

[85]:
```
        DeviceType  DeviceAgeMonths
1           Lights        29.912603
0           Camera        30.035422
2  Security System        30.485955
4       Thermostat        30.544755
```

```
3    Smart Speaker        30.593863
```

[86]:
```
x.groupby(["DeviceType"])[["UserPreferences"]].mean().reset_index().
 ↪sort_values(by="UserPreferences",ascending=True)
```

[86]:
```
         DeviceType  UserPreferences
4       Thermostat         0.502406
2  Security System         0.503745
1           Lights         0.514259
0           Camera         0.518619
3    Smart Speaker         0.518953
```

[87]:
```
x.groupby(["DeviceType"])[["SmartHomeEfficiency"]].mean().reset_index().
 ↪sort_values(by="SmartHomeEfficiency",ascending=True)
```

[87]:
```
         DeviceType  SmartHomeEfficiency
4       Thermostat             0.351299
2  Security System             0.375468
1           Lights             0.379945
0           Camera             0.386921
3    Smart Speaker             0.388087
```

[ ]:

[ ]:

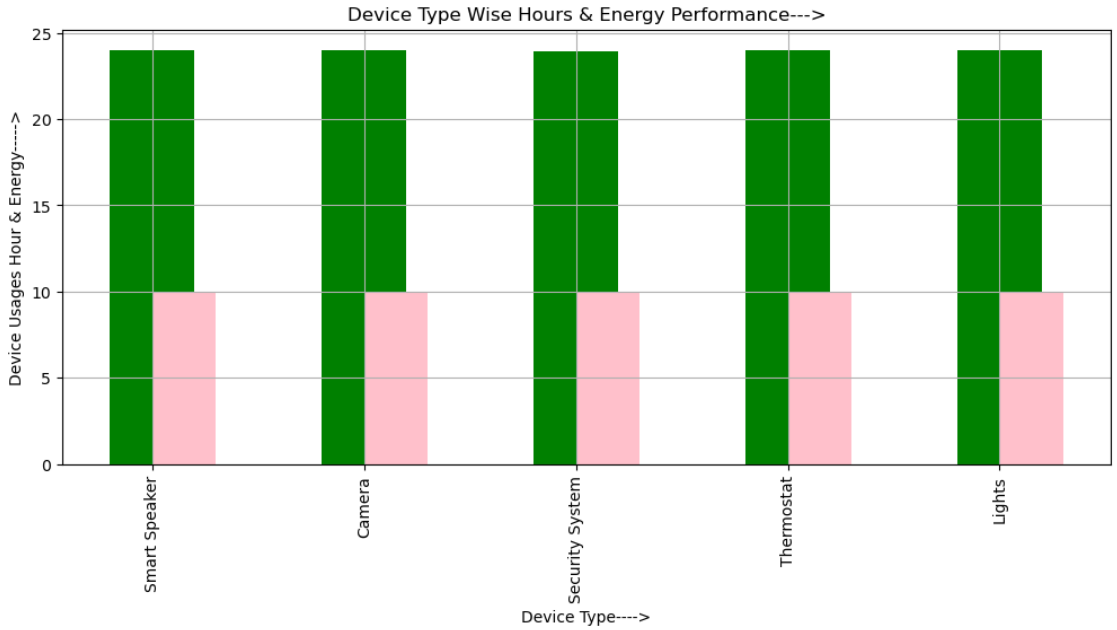
[ ]:

[88]:
```
h=x.groupby(["DeviceType"]).agg({"UsageHoursPerDay":
 ↪["min","max"],"EnergyConsumption":["min","max"],"DeviceType":"count"}).
 ↪reset_index()
h
```

[88]:
```
         DeviceType UsageHoursPerDay            EnergyConsumption            \
                                 min        max                 min       max
0           Camera         0.505335  23.984732            0.113203  9.998071
1           Lights         0.505664  23.967507            0.102385  9.996765
2  Security System         0.509642  23.945298            0.101562  9.988564
3    Smart Speaker         0.509436  23.987326            0.109336  9.995101
4       Thermostat         0.501241  23.981372            0.113067  9.987115

  DeviceType
       count
0       1101
1       1087
2       1068
3       1108
```

```
4        1039
```

[89]:
```
plt.figure(figsize=(12,5))
plt.bar(x["DeviceType"],x["UsageHoursPerDay"],color="g",width=0.
  ↪4,align="center",label="Usages Hour Day")
plt.bar(x["DeviceType"],x["EnergyConsumption"],color="pink",width=0.
  ↪3,align="edge",label="Energy")
plt.xlabel("Device Type---->")
plt.ylabel("Device Usages Hour & Energy----->")
plt.xticks(rotation=90)
plt.title("Device Type Wise Hours & Energy Performance--->")
plt.grid()
plt.show()
```



2 Device Type wize Usages Hour & Energy confirm = Camera

[91]:
```
t=x.loc[(x["DeviceType"]=="Camera")]
t.loc[(t["UsageHoursPerDay"].between(0,24))]
```

[91]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	UserPreferences	\
1	2	Camera	19.973343	8.610689	1	
3	4	Camera	7.011127	2.341653	0	
4	5	Camera	22.610684	4.859069	1	
9	10	Camera	17.468553	7.212756	1	
12	13	Camera	12.632658	7.169462	1	

```
…      …         …                 …                  …                 …
5365    5366     Camera           9.616876           5.277574                 0
5377    5378     Camera          13.517631           1.438874                 1
5379    5380     Camera           7.209839           3.493130                 0
5391    5392     Camera          13.001725           0.671065                 1
5396    5397     Camera          19.301279           0.792446                 1

      MalfunctionIncidents  DeviceAgeMonths  SmartHomeEfficiency
1                        0               29                    1
3                        3               15                    0
4                        3               36                    1
9                        4               58                    0
12                       1                9                    1
…                      …                 …                    …
5365                     1               29                    1
5377                     0               58                    0
5379                     2               35                    1
5391                     1               30                    0
5396                     1               33                    1

[1101 rows x 8 columns]
```

[92]:
```
t.loc[(x["EnergyConsumption"].between(0,10))]
```

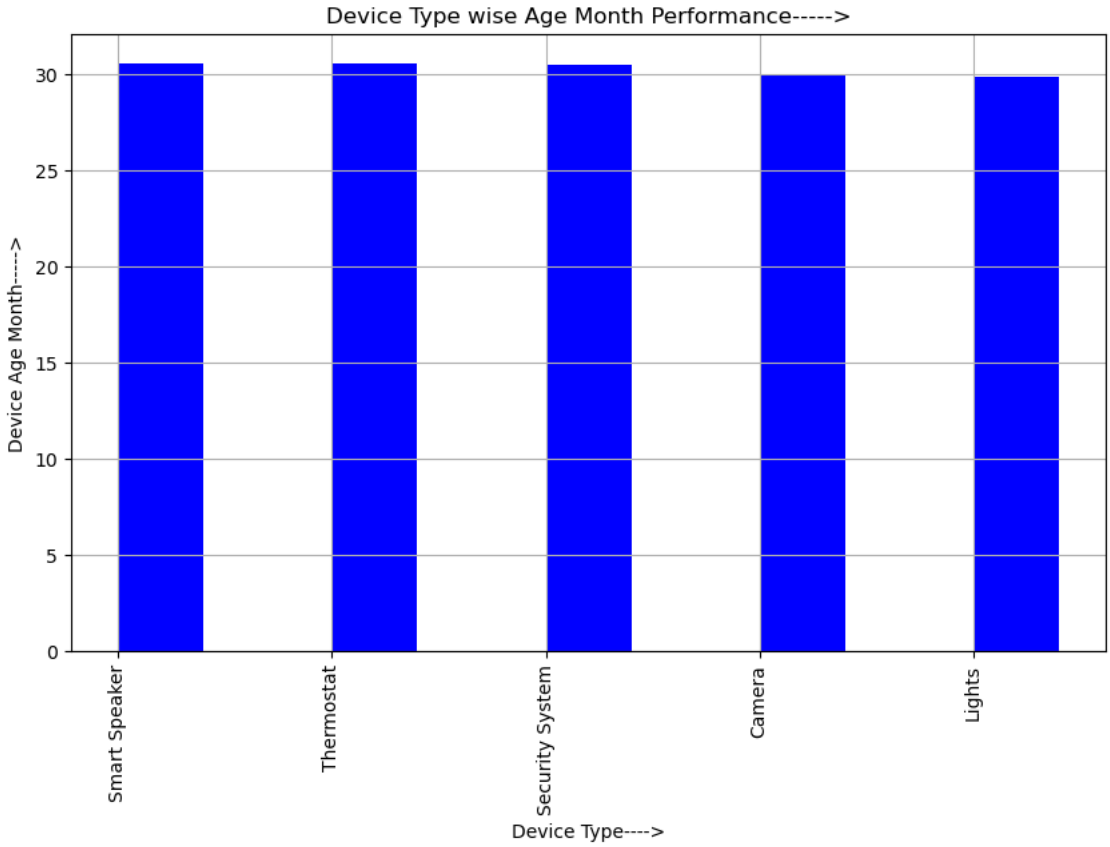
[92]:
```
      UserID DeviceType  UsageHoursPerDay  EnergyConsumption  UserPreferences  \
1          2     Camera         19.973343           8.610689                1
3          4     Camera          7.011127           2.341653                0
4          5     Camera         22.610684           4.859069                1
9         10     Camera         17.468553           7.212756                1
12        13     Camera         12.632658           7.169462                1
…       …         …                 …                  …                 …
5365    5366     Camera          9.616876           5.277574                 0
5377    5378     Camera         13.517631           1.438874                 1
5379    5380     Camera          7.209839           3.493130                 0
5391    5392     Camera         13.001725           0.671065                 1
5396    5397     Camera         19.301279           0.792446                 1

      MalfunctionIncidents  DeviceAgeMonths  SmartHomeEfficiency
1                        0               29                    1
3                        3               15                    0
4                        3               36                    1
9                        4               58                    0
12                       1                9                    1
…                      …                 …                    …
5365                     1               29                    1
5377                     0               58                    0
5379                     2               35                    1
```

```
5391                    1               30                    0
5396                    1               33                    1

[1101 rows x 8 columns]
```

[93]:
```
m=x.groupby(["DeviceType"])[["DeviceAgeMonths"]].mean().reset_index().
  ↪sort_values(by="DeviceAgeMonths",ascending=False)
m
```

[93]:
```
         DeviceType  DeviceAgeMonths
3    Smart Speaker        30.593863
4       Thermostat        30.544755
2  Security System        30.485955
0           Camera        30.035422
1           Lights        29.912603
```

[94]:
```
plt.figure(figsize=(10,6))
plt.bar(m["DeviceType"],m["DeviceAgeMonths"],color="b",width=0.
  ↪4,align="edge",label="Device Age Month")
plt.title("Device Type wise Age Month Performance----->")
plt.xlabel("Device Type---->")
plt.ylabel("Device Age Month----->")
plt.grid()
plt.xticks(rotation=90)
plt.show()
```



3 Devise Type wise Device Age Month confirm = 30 % MAX

[96]:
```
d=x.loc[(x["DeviceAgeMonths"]==30)]
d
```

[96]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	\
13	14	Thermostat	14.145163	7.385760	
56	57	Smart Speaker	12.799932	6.377145	
65	66	Smart Speaker	1.702030	4.702013	
136	137	Smart Speaker	9.884316	8.494920	
152	153	Lights	13.391828	9.339436	
…	…	…	…	…	
5170	5171	Smart Speaker	21.185917	8.206446	
5274	5275	Thermostat	13.368882	7.631937	
5320	5321	Thermostat	20.996806	6.489874	
5391	5392	Camera	13.001725	0.671065	
5402	5403	Thermostat	13.540381	9.043076	

	UserPreferences	MalfunctionIncidents	DeviceAgeMonths	\

```
13                  0                   2               30
56                  1                   3               30
65                  0                   1               30
136                 0                   2               30
152                 0                   3               30
…                   …                   …               …
5170                0                   4               30
5274                1                   3               30
5320                0                   3               30
5391                1                   1               30
5402                0                   0               30

      SmartHomeEfficiency
13                      0
56                      1
65                      0
136                     0
152                     0
…                       …
5170                    0
5274                    1
5320                    0
5391                    0
5402                    0

[97 rows x 8 columns]
```

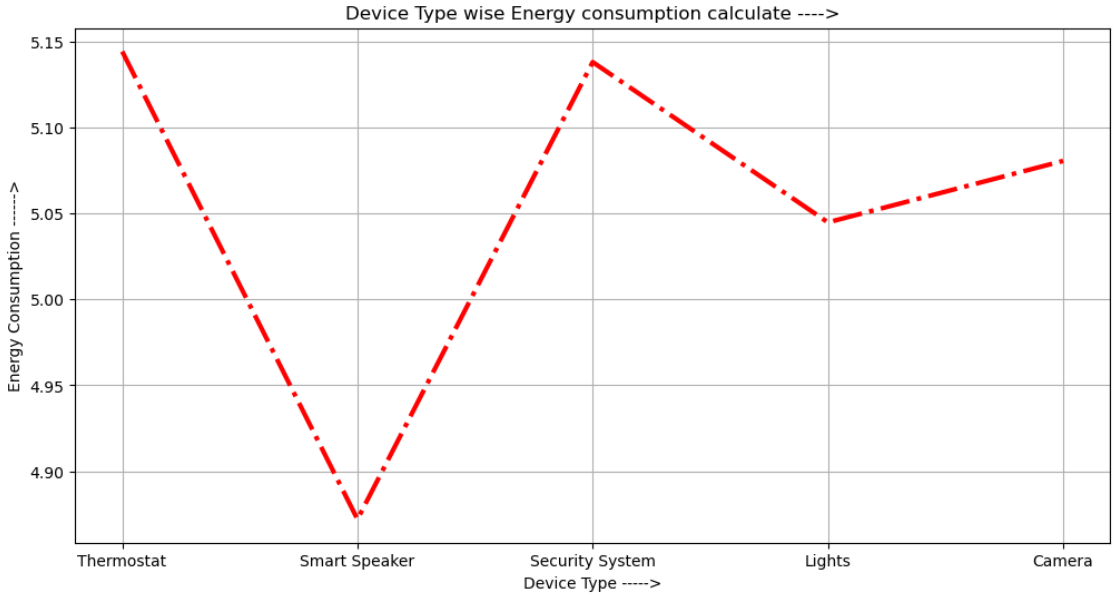
[97]:
```
z=x.groupby(["DeviceType"])[["EnergyConsumption"]].mean().reset_index().
  ↪sort_values(by="DeviceType",ascending=False)
z
```

[97]:
```
         DeviceType  EnergyConsumption
4        Thermostat           5.144356
3     Smart Speaker           4.872036
2  Security System            5.138192
1            Lights           5.044884
0            Camera           5.080666
```

3.1 Device Type Final = Camera having Less Energy Consumption

[99]:
```
plt.figure(figsize=(12,6))
plt.plot(z["DeviceType"],z["EnergyConsumption"],color="r",lw=3,ls="-.
  ↪",markersize=6,label="Energy Consu")
plt.title("Device Type wise Energy consumption calculate ---->")
plt.xlabel("Device Type ----->")
plt.ylabel("Energy Consumption ------>")
plt.grid()
```

```
plt.show()
```



[100]:
```
x.groupby(["DeviceType"])[["UserPreferences","UsageHoursPerDay"]].mean().
↪reset_index()
```

[100]:

	DeviceType	UserPreferences	UsageHoursPerDay
0	Camera	0.518619	12.113435
1	Lights	0.514259	12.052646
2	Security System	0.503745	12.016149
3	Smart Speaker	0.518953	11.979308
4	Thermostat	0.502406	12.105753

3.2 device type Final = Camera having more user Preferences

[102]:
```
x.groupby(["DeviceType"])[["MalfunctionIncidents","DeviceAgeMonths"]].mean().
↪reset_index()
```

[102]:

	DeviceType	MalfunctionIncidents	DeviceAgeMonths
0	Camera	2.087193	30.035422
1	Lights	2.021159	29.912603
2	Security System	2.048689	30.485955
3	Smart Speaker	2.037906	30.593863
4	Thermostat	2.140520	30.544755

3.3 Device Type confirm =Smart Speaker because device Age month more

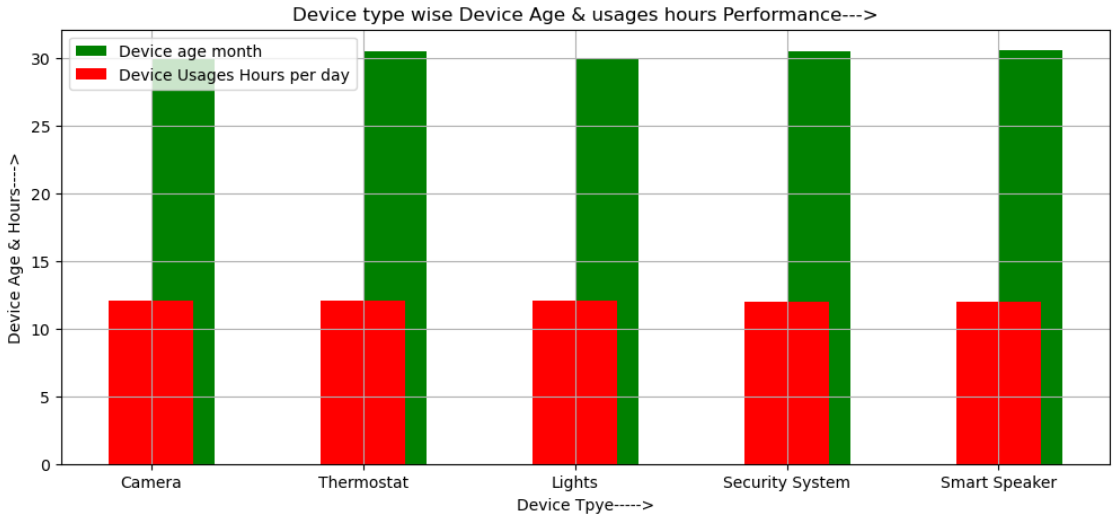
[104]:
```
s=x.groupby(["DeviceType"]).agg({"DeviceAgeMonths":"mean","UsageHoursPerDay":
 ↪"mean"}).reset_index().sort_values(by="UsageHoursPerDay",ascending=False)
s
```

[104]:

	DeviceType	DeviceAgeMonths	UsageHoursPerDay
0	Camera	30.035422	12.113435
4	Thermostat	30.544755	12.105753
1	Lights	29.912603	12.052646
2	Security System	30.485955	12.016149
3	Smart Speaker	30.593863	11.979308

3.3.1 Smart Speaker Final = because device age month more & Usages Hour Per Day Less

[106]:
```
plt.figure(figsize=(12,5))
plt.bar(s["DeviceType"],s["DeviceAgeMonths"],color="g",width=0.
 ↪3,align="edge",label="Device age month")
plt.bar(s["DeviceType"],s["UsageHoursPerDay"],color="r",width=0.
 ↪4,align="center",label="Device Usages Hours per day")
plt.title("Device type wise Device Age & usages hours Performance--->")
plt.xlabel("Device Tpye----->")
plt.ylabel("Device Age & Hours---->")
plt.grid()
plt.legend()
plt.show()
```



[107]:
```
d.loc[(x["DeviceAgeMonths"]==31)]
```

[107]:
```
Empty DataFrame
Columns: [UserID, DeviceType, UsageHoursPerDay, EnergyConsumption,
UserPreferences, MalfunctionIncidents, DeviceAgeMonths, SmartHomeEfficiency]
Index: []
```

[108]:
```
x["MalfunctionIncidents"].unique()
```

[108]:
```
array([4, 0, 3, 2, 1], dtype=int64)
```

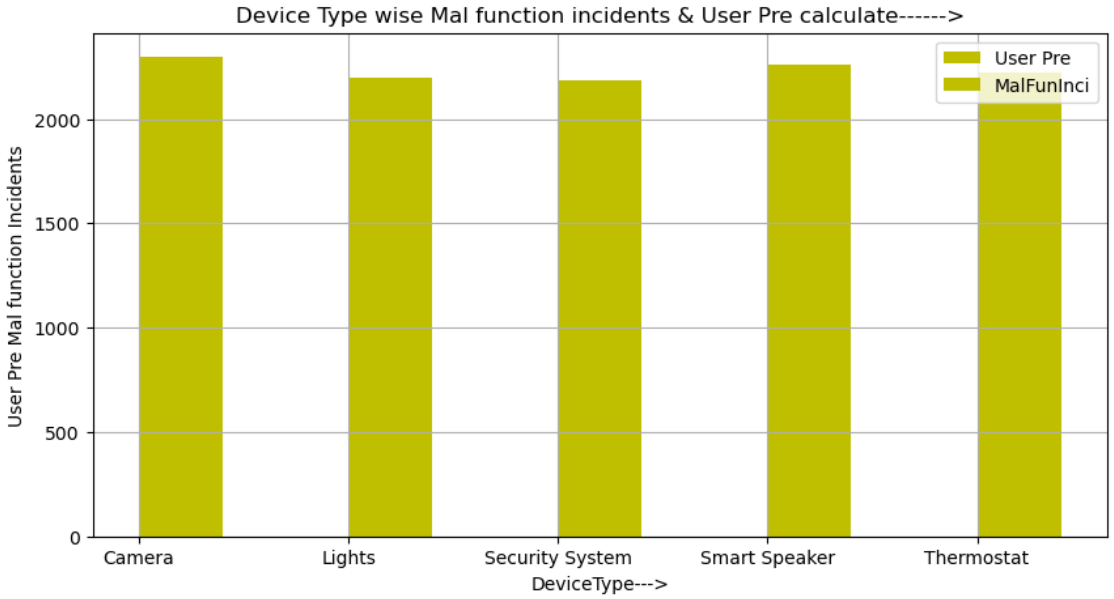
[109]:
```
a=x.groupby(["DeviceType"])[["UserPreferences","MalfunctionIncidents"]].sum().
  ↪reset_index()
a
```

[109]:

	DeviceType	UserPreferences	MalfunctionIncidents
0	Camera	571	2298
1	Lights	559	2197
2	Security System	538	2188
3	Smart Speaker	575	2258
4	Thermostat	522	2224

4 device Final = Thermostat because MalfunctionIncidents are Less

[111]:
```
plt.figure(figsize=(10,5))
plt.bar(a["DeviceType"],a["UserPreferences"],color="y",width=0.
  ↪4,align="edge",label="User Pre")
plt.bar(a["DeviceType"],a["MalfunctionIncidents"],color="y",width=0.
  ↪4,align="edge",label="MalFunInci")
plt.title("Device Type wise Mal function incidents & User Pre calculate------>")
plt.xlabel("DeviceType--->")
plt.ylabel("User Pre Mal function Incidents")
plt.legend()
plt.grid()
plt.show()
```



[112]:
```
y=x.loc[(x["DeviceType"]=="Thermostat")]
y.loc[(x["MalfunctionIncidents"].between(0,2))]
```

[112]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	\
5	6	Thermostat	3.422127	5.038625	
13	14	Thermostat	14.145163	7.385760	
15	16	Thermostat	16.297891	9.665162	
27	28	Thermostat	9.452368	0.281722	
43	44	Thermostat	18.933529	7.707831	
…	…	…	…	…	
5359	5360	Thermostat	1.616667	9.457098	
5375	5376	Thermostat	7.856096	1.091419	
5388	5389	Thermostat	13.472427	6.728036	
5398	5399	Thermostat	4.556314	5.871764	
5402	5403	Thermostat	13.540381	9.043076	

	UserPreferences	MalfunctionIncidents	DeviceAgeMonths	\
5	1	0	3	
13	0	2	30	
15	1	0	38	
27	1	0	47	
43	0	0	18	
…	…	…	…	
5359	1	0	54	
5375	0	2	8	
5388	1	2	2	
5398	1	0	28	

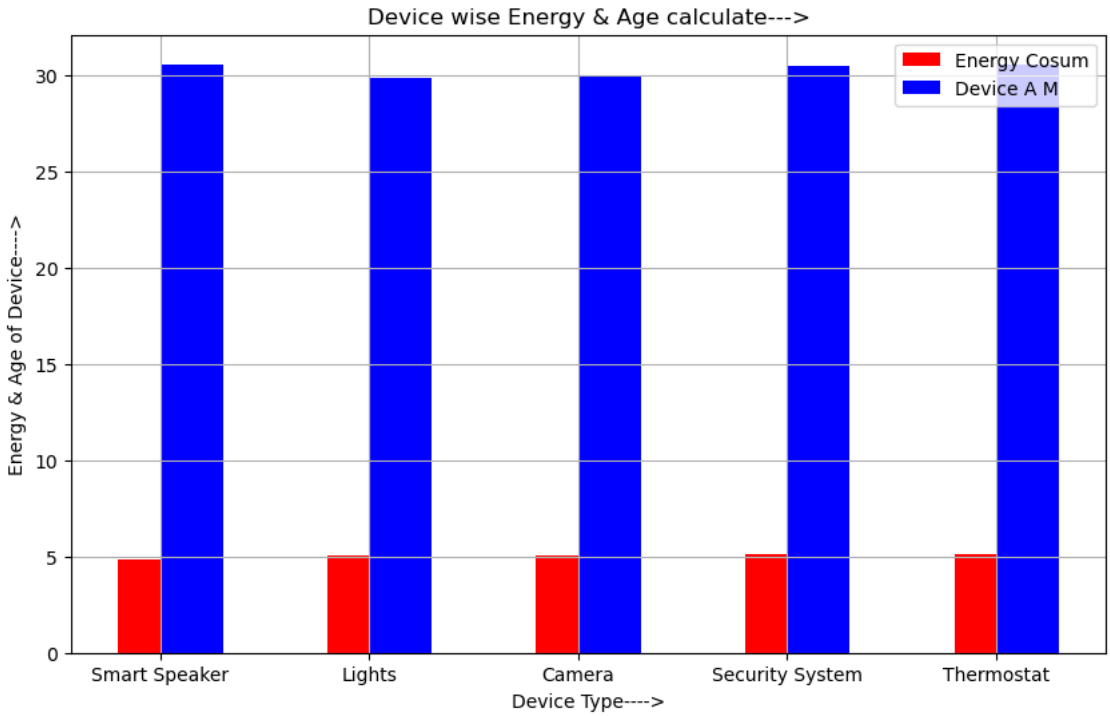
```
5402                0                     0               30

      SmartHomeEfficiency
5                       1
13                      0
15                      1
27                      1
43                      0
…                     …
5359                    0
5375                    1
5388                    1
5398                    0
5402                    0

[571 rows x 8 columns]
```

[113]:
```
ss=x.groupby(["DeviceType"])[["EnergyConsumption","DeviceAgeMonths"]].mean().
 ↪reset_index().sort_values(by="EnergyConsumption",ascending=True)
ss
```

[113]:

	DeviceType	EnergyConsumption	DeviceAgeMonths
3	Smart Speaker	4.872036	30.593863
1	Lights	5.044884	29.912603
0	Camera	5.080666	30.035422
2	Security System	5.138192	30.485955
4	Thermostat	5.144356	30.544755

[114]:
```
plt.figure(figsize=(10,6))
plt.bar(ss["DeviceType"],ss["EnergyConsumption"],color="r",width=0.
 ↪4,align="center",label="Energy Cosum")
plt.bar(ss["DeviceType"],ss["DeviceAgeMonths"],color="b",width=0.
 ↪3,align="edge",label="Device A M")
plt.xlabel("Device Type---->")
plt.ylabel("Energy & Age of Device---->")
plt.title("Device wise Energy & Age calculate--->")
plt.legend()
plt.grid()
plt.show()
```



5 Device Type Final = Lights because Energy Consumption is Less

[116]:
```
q=x.groupby(["DeviceType","DeviceAgeMonths"])[["EnergyConsumption"]].mean().
  ↪reset_index().sort_values(by="EnergyConsumption",ascending=True)
q
```

[116]:

	DeviceType	DeviceAgeMonths	EnergyConsumption
133	Security System	16	2.996148
105	Lights	47	3.068667
221	Smart Speaker	45	3.259841
162	Security System	45	3.307480
93	Lights	35	3.337385
..	…	…	…
247	Thermostat	12	6.579902
76	Lights	18	6.587448
124	Security System	7	6.652444
33	Camera	34	6.795095
242	Thermostat	7	6.838405

[295 rows x 3 columns]

[117]:
```
y=x.loc[(x["DeviceType"]=="Security System")]
j=y.loc[(y["DeviceAgeMonths"]==16)]
j
```

[117]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	UserPreferences	MalfunctionIncidents	DeviceAgeMonths	SmartHomeEfficiency
154	155	Security System	12.710874	6.960421	0	2	16	0
167	168	Security System	16.226647	3.785228	1	1	16	1
244	245	Security System	0.973239	1.745385	1	1	16	1
689	690	Security System	3.305454	4.968932	0	4	16	0
1503	1504	Security System	4.413655	0.535319	1	2	16	1
2001	2002	Security System	12.859745	1.262875	1	2	16	1
2306	2307	Security System	1.354008	0.819555	1	0	16	
3602	3603	Security System	12.067080	1.946923	1	2	16	
3821	3822	Security System	20.882834	4.102201	1	2	16	
3910	3911	Security System	7.864141	4.881238	1	0	16	
3985	3986	Security System	2.166530	0.948076	1	0	16	
4177	4178	Security System	17.433781	0.233096	1	1	16	
4204	4205	Security System	15.814475	4.788043	0	4	16	
4548	4549	Security System	3.422115	1.601503	1	4	16	
5037	5038	Security System	1.869984	5.554348	0	1	16	
5273	5274	Security System	6.945153	3.805218	1	2	16	

```
2306                    1
3602                    1
3821                    1
3910                    1
3985                    1
4177                    1
4204                    0
4548                    1
5037                    0
5273                    0
```

6 Device Type confirm = Security System because low Energy Consumption

[119]:
```
y=pd.get_dummies(x["DeviceType"],drop_first=False).replace({True:1,False:0})
y
```

[119]:
```
      Camera  Lights  Security System  Smart Speaker  Thermostat
0          0       0                0              1           0
1          1       0                0              0           0
2          0       0                1              0           0
3          1       0                0              0           0
4          1       0                0              0           0
…        …     …                …              …           …
5398       0       0                0              0           1
5399       0       1                0              0           0
5400       0       0                0              1           0
5401       0       0                1              0           0
5402       0       0                0              0           1

[5403 rows x 5 columns]
```

[120]:
```
F=pd.concat([x,y],axis=1)
F
```

[120]:
```
      UserID       DeviceType  UsageHoursPerDay  EnergyConsumption  \
0          1    Smart Speaker         15.307188           1.961607
1          2           Camera         19.973343           8.610689
2          3  Security System         18.911535           2.651777
3          4           Camera          7.011127           2.341653
4          5           Camera         22.610684           4.859069
…        …              …                …                  …
5398    5399       Thermostat          4.556314           5.871764
5399    5400           Lights          0.561856           1.555992
5400    5401    Smart Speaker         11.096236           7.677779
5401    5402  Security System          8.782169           7.467929
```

```
5402    5403       Thermostat         13.540381           9.043076

      UserPreferences  MalfunctionIncidents  DeviceAgeMonths  \
0                   1                     4               36
1                   1                     0               29
2                   1                     0               20
3                   0                     3               15
4                   1                     3               36
…                 …                     …               …
5398                1                     0               28
5399                1                     4               24
5400                0                     0               42
5401                0                     2               28
5402                0                     0               30

      SmartHomeEfficiency  Camera  Lights  Security System  Smart Speaker  \
0                       1       0       0                0              1
1                       1       1       0                0              0
2                       1       0       0                1              0
3                       0       1       0                0              0
4                       1       1       0                0              0
…                     …     …     …               …            …
5398                    0       0       0                0              0
5399                    0       0       1                0              0
5400                    0       0       0                0              1
5401                    1       0       0                1              0
5402                    0       0       0                0              0

      Thermostat
0              0
1              0
2              0
3              0
4              0
…            …
5398           1
5399           0
5400           0
5401           0
5402           1

[5403 rows x 13 columns]
```

[121]:
```
F.drop(columns="DeviceType",inplace=True)
F
```

[121]:

	UserID	UsageHoursPerDay	EnergyConsumption	UserPreferences	\
0	1	15.307188	1.961607	1	
1	2	19.973343	8.610689	1	
2	3	18.911535	2.651777	1	
3	4	7.011127	2.341653	0	
4	5	22.610684	4.859069	1	
…	…	…	…	…	
5398	5399	4.556314	5.871764	1	
5399	5400	0.561856	1.555992	1	
5400	5401	11.096236	7.677779	0	
5401	5402	8.782169	7.467929	0	
5402	5403	13.540381	9.043076	0	

	MalfunctionIncidents	DeviceAgeMonths	SmartHomeEfficiency	Camera	\
0	4	36	1	0	
1	0	29	1	1	
2	0	20	1	0	
3	3	15	0	1	
4	3	36	1	1	
…	…	…	…	…	
5398	0	28	0	0	
5399	4	24	0	0	
5400	0	42	0	0	
5401	2	28	1	0	
5402	0	30	0	0	

	Lights	Security System	Smart Speaker	Thermostat
0	0	0	1	0
1	0	0	0	0
2	0	1	0	0
3	0	0	0	0
4	0	0	0	0
…	…	…	…	…
5398	0	0	0	1
5399	1	0	0	0
5400	0	0	1	0
5401	0	1	0	0
5402	0	0	0	1

[5403 rows x 12 columns]

[122]:
```
F.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5403 entries, 0 to 5402
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
```

```
---  ------                 --------------  -----
 0   UserID                 5403 non-null   int64
 1   UsageHoursPerDay       5403 non-null   float64
 2   EnergyConsumption      5403 non-null   float64
 3   UserPreferences        5403 non-null   int64
 4   MalfunctionIncidents   5403 non-null   int64
 5   DeviceAgeMonths        5403 non-null   int64
 6   SmartHomeEfficiency    5403 non-null   int64
 7   Camera                 5403 non-null   int64
 8   Lights                 5403 non-null   int64
 9   Security System        5403 non-null   int64
 10  Smart Speaker          5403 non-null   int64
 11  Thermostat             5403 non-null   int64
dtypes: float64(2), int64(10)
memory usage: 506.7 KB
```

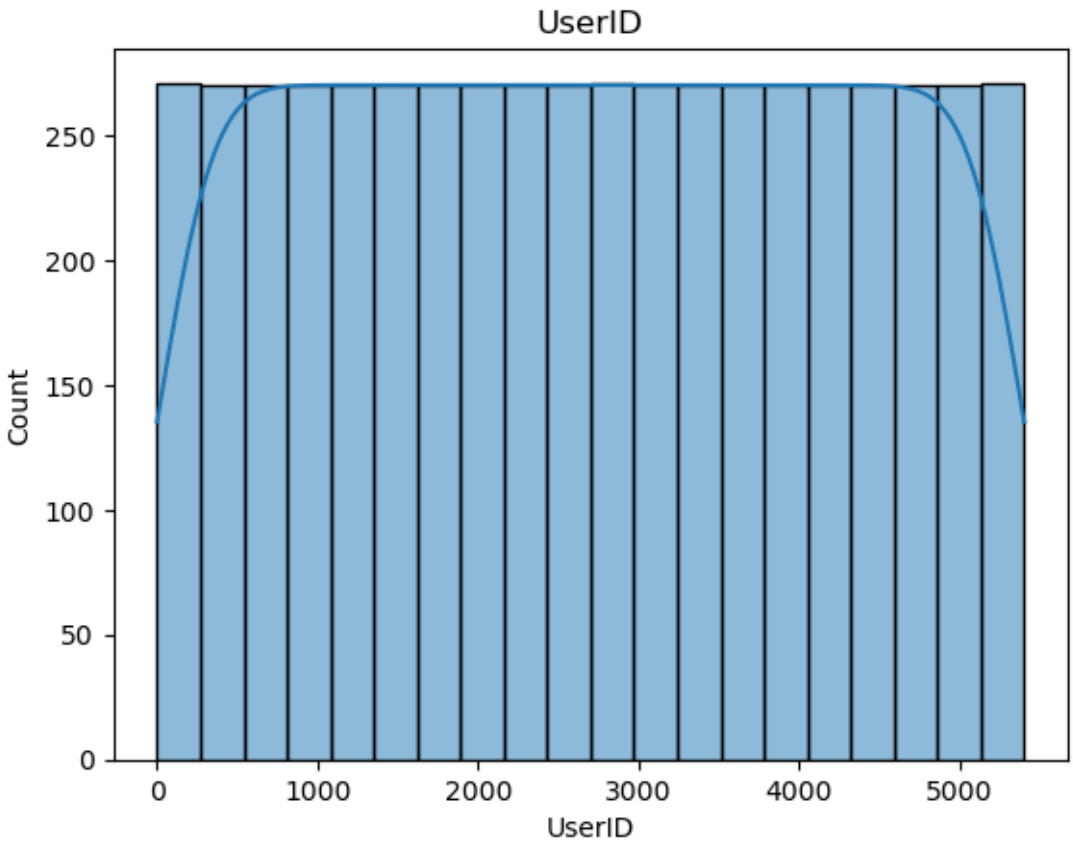
[123]:
```
F.columns
```

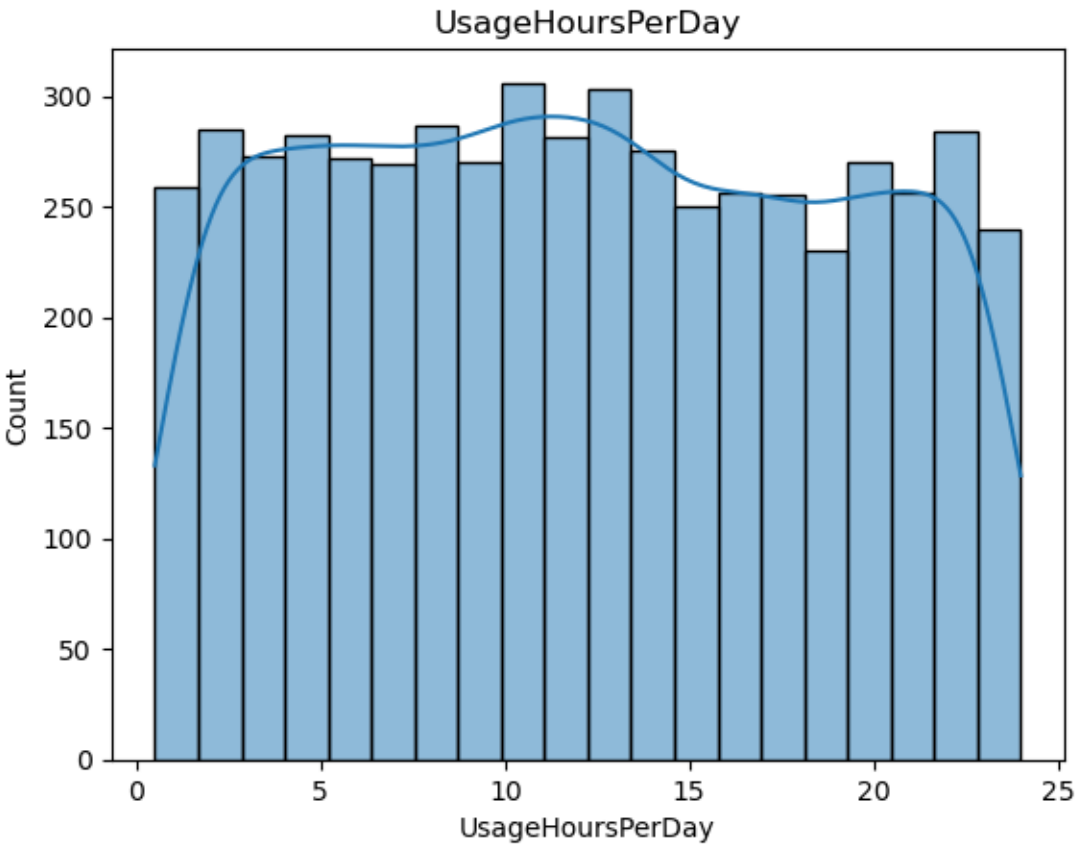
[123]:
```
Index(['UserID', 'UsageHoursPerDay', 'EnergyConsumption', 'UserPreferences',
       'MalfunctionIncidents', 'DeviceAgeMonths', 'SmartHomeEfficiency',
       'Camera', 'Lights', 'Security System', 'Smart Speaker', 'Thermostat'],
      dtype='object')
```

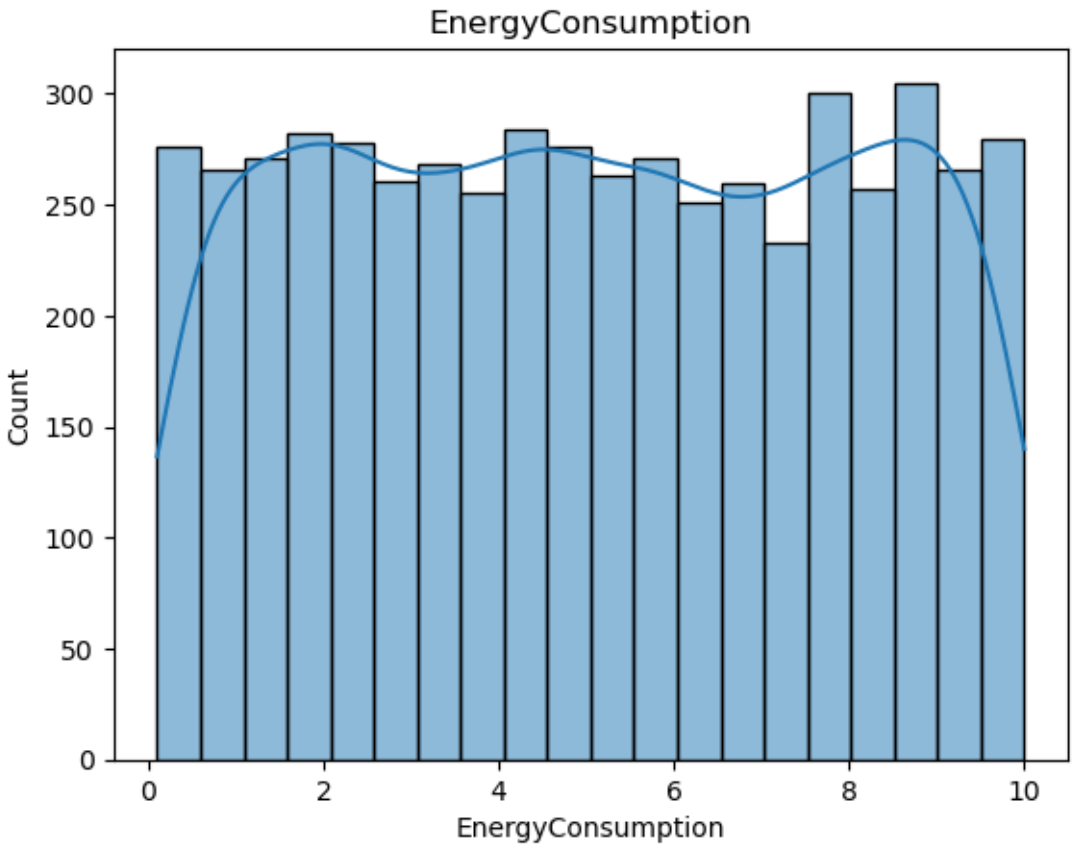
[124]:
```
F["UserID"].unique()
```

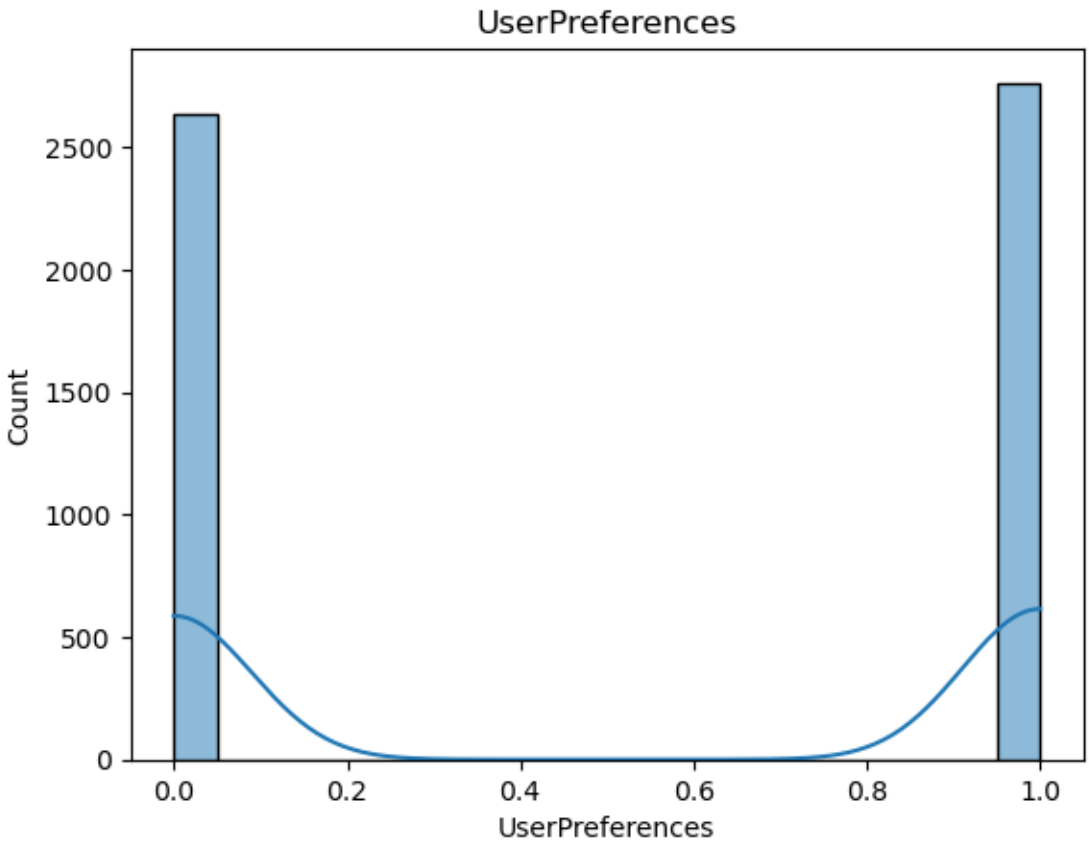
[124]:
```
array([   1,    2,    3, …, 5401, 5402, 5403], dtype=int64)
```

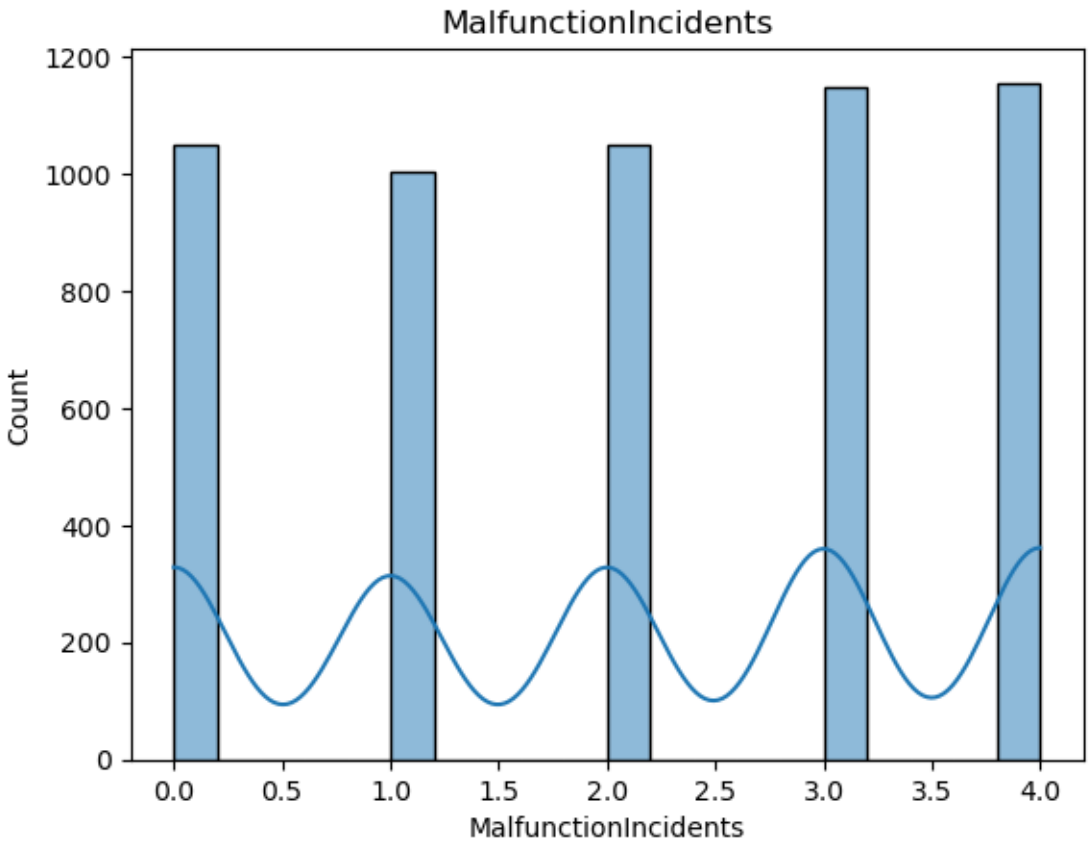
[125]:
```
import matplotlib.pyplot as plt
import seaborn as sns
for i in F.columns:
    sns.histplot(F[i],bins=20,kde=True)
    plt.title(i)
    plt.show()
```

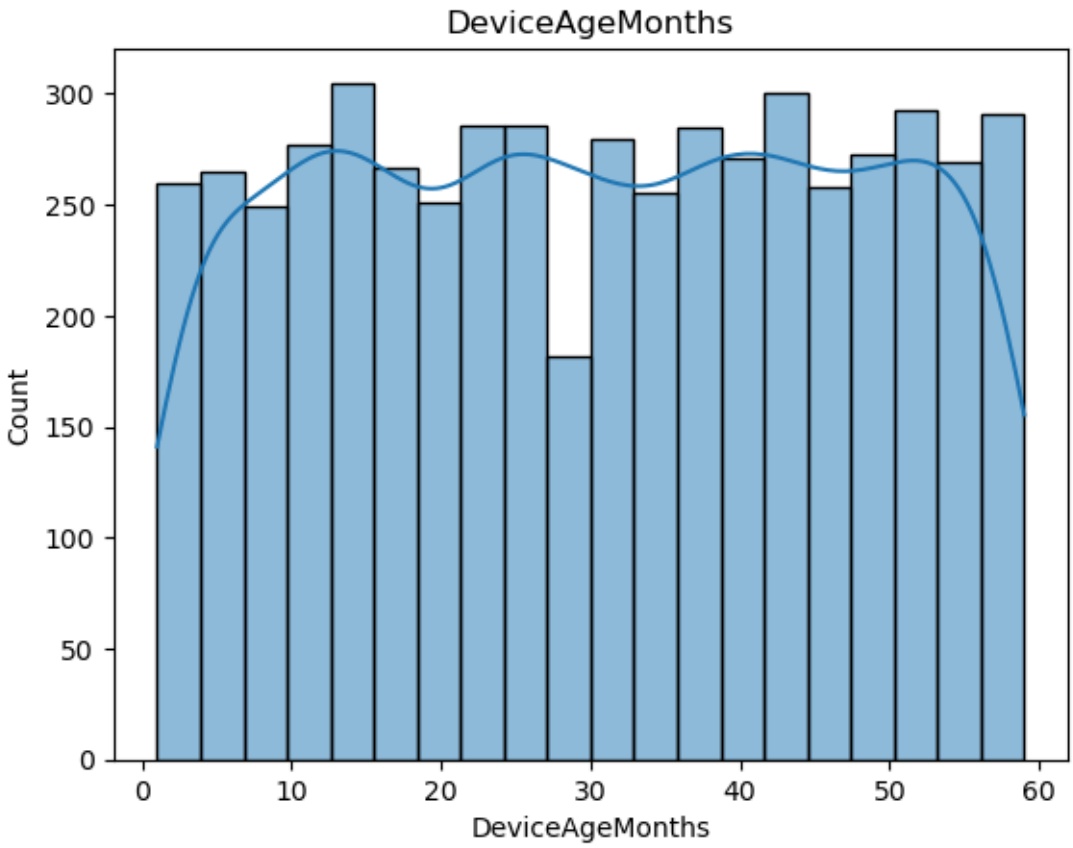

UserID
250
200
150
100
50
0
Count
0
1000
2000
3000
4000
5000
UserID

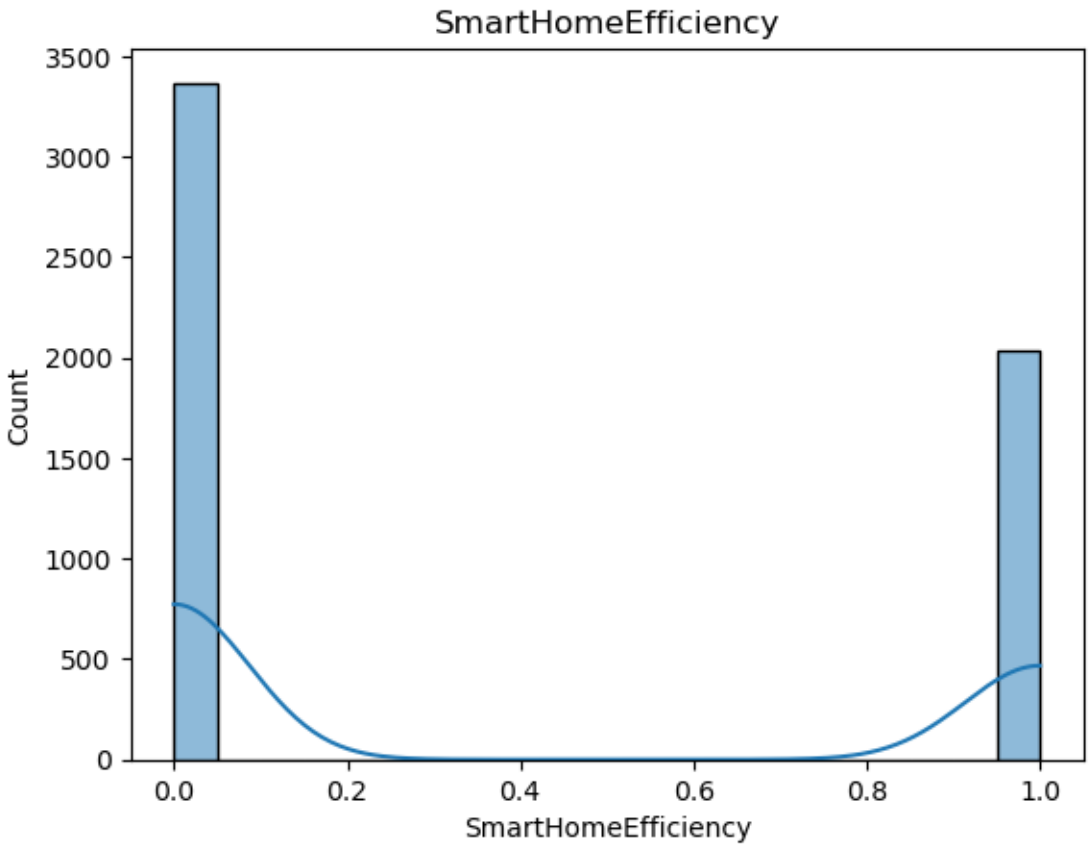

UsageHoursPerDay
Count
300
250
200
150
100
50
0
0
5
10
15
20
25
UsageHoursPerDay

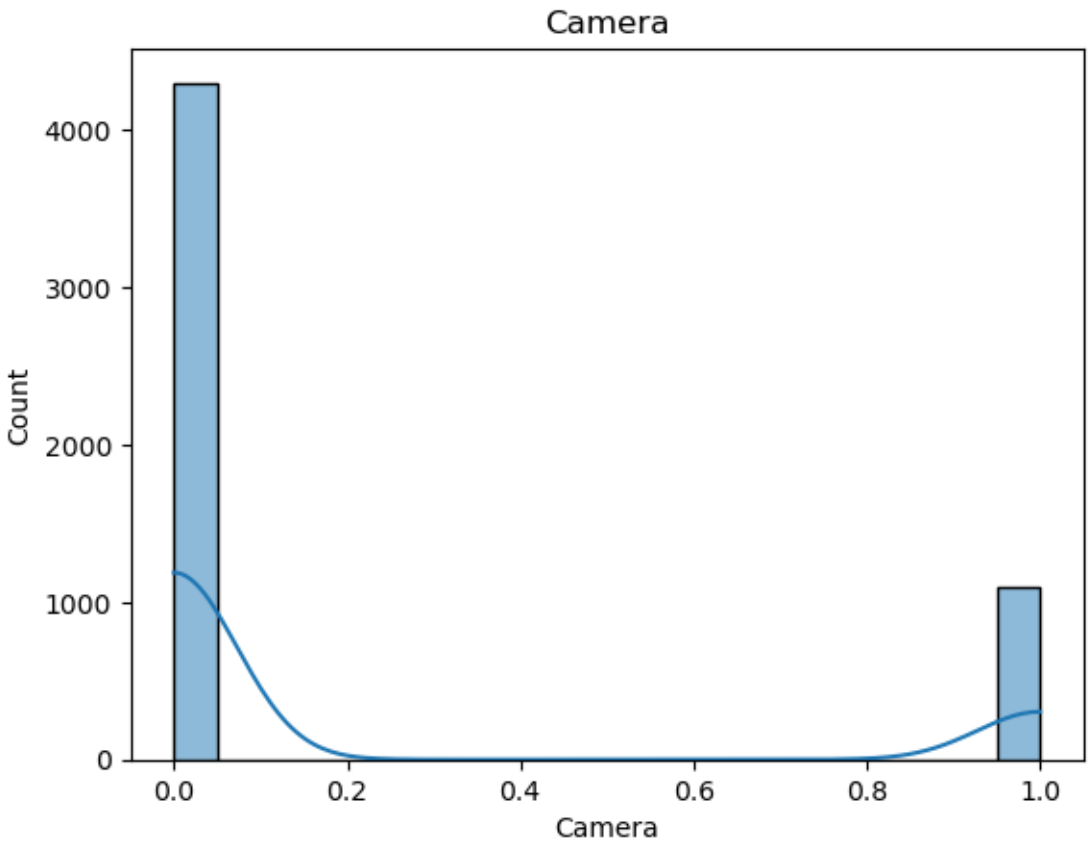

EnergyConsumption
Count
300
250
200
150
100
50
0
0
2
4
6
8
10
EnergyConsumption

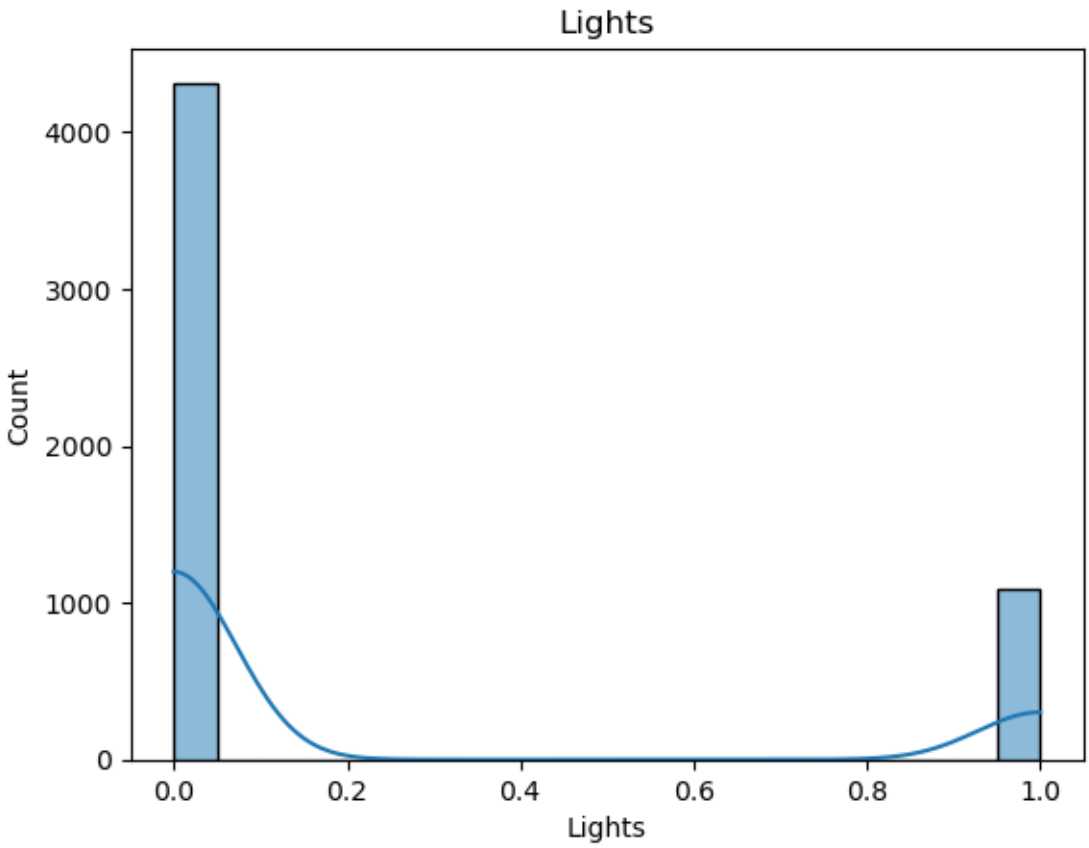

UserPreferences
2500
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
UserPreferences

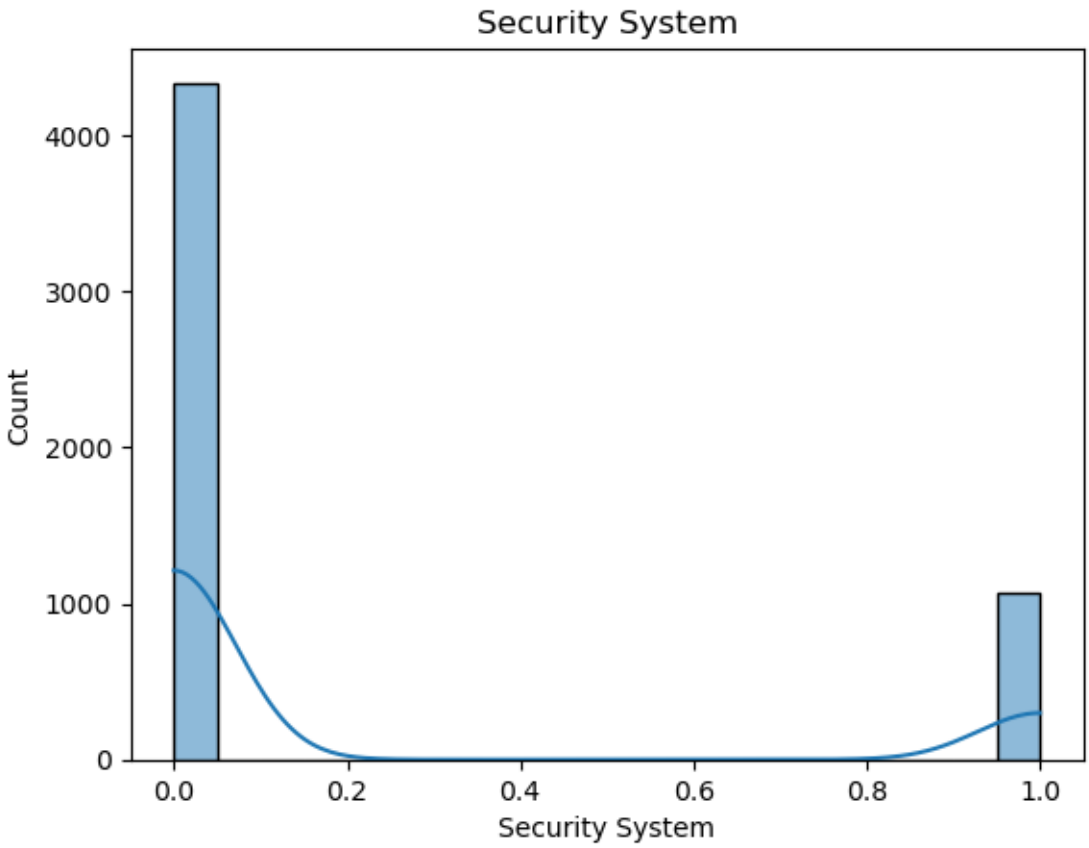

MalfunctionIncidents
1200
1000
800
600
400
200
0
Count
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
MalfunctionIncidents

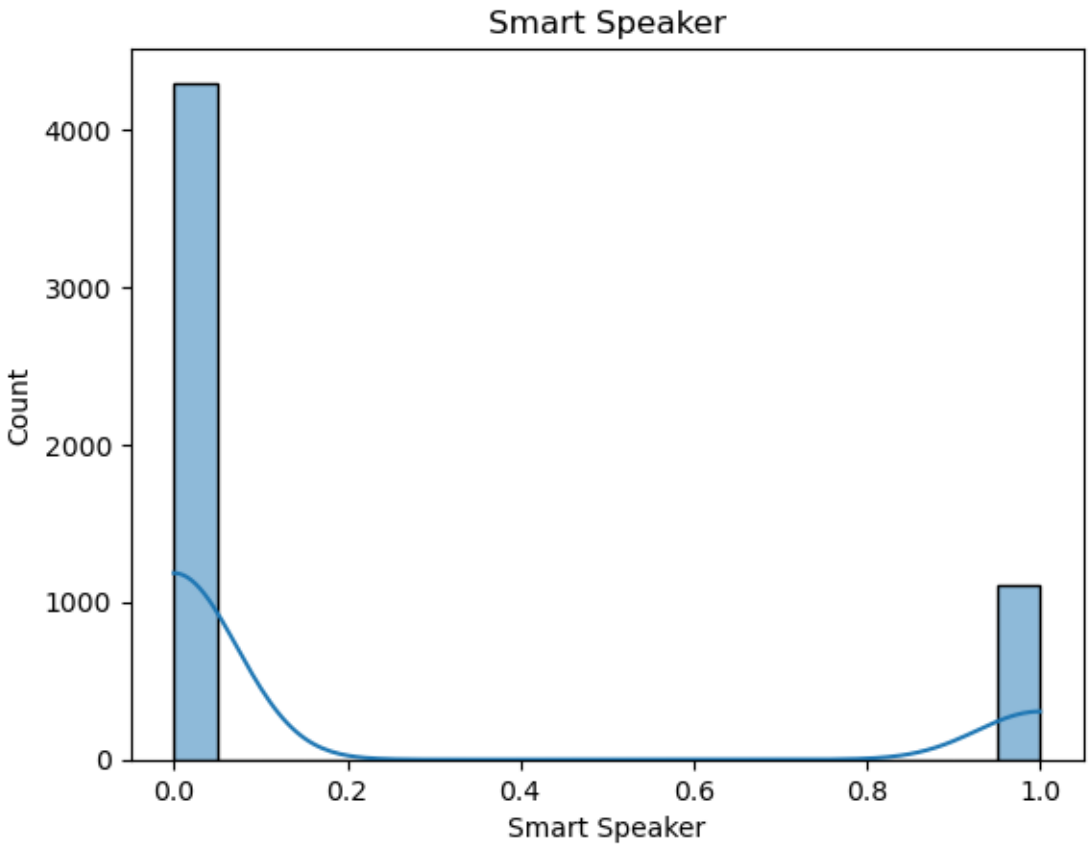

DeviceAgeMonths
Count
DeviceAgeMonths
0
50
100
150
200
250
300
0
10
20
30
40
50
60

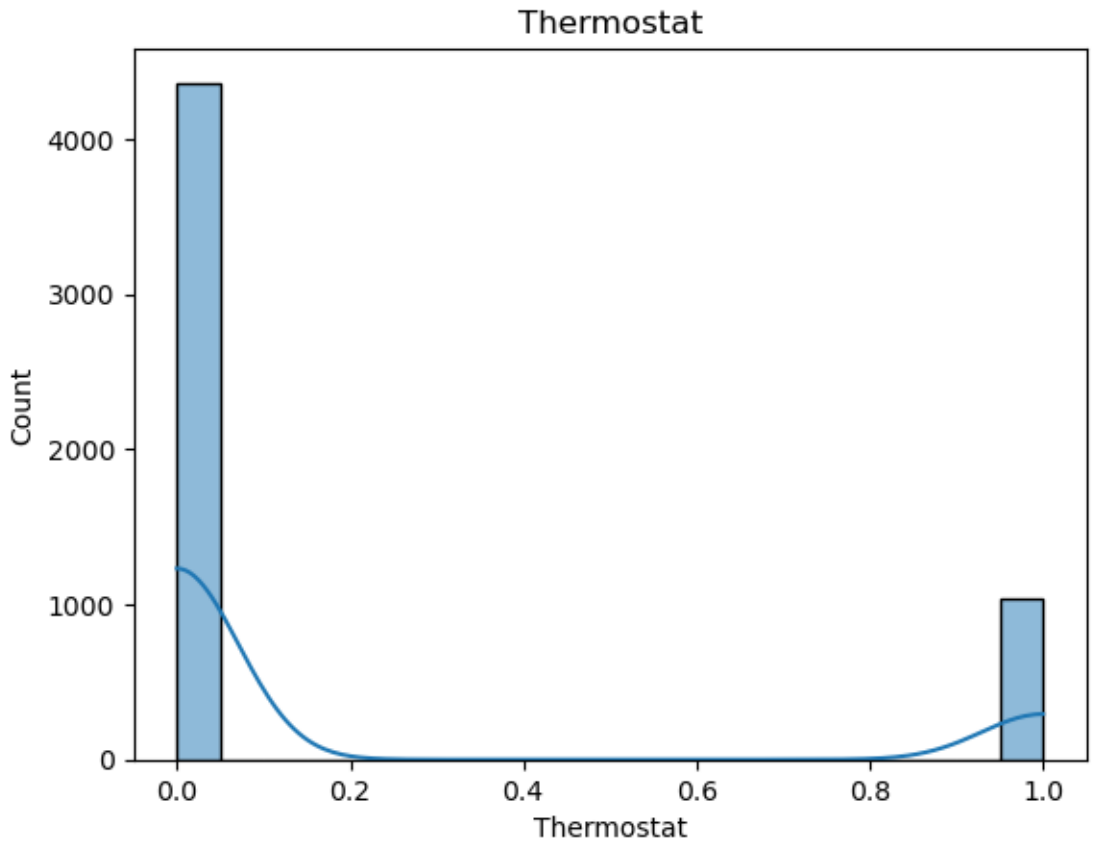

SmartHomeEfficiency
3500
3000
2500
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
SmartHomeEfficiency


Camera
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Camera


Lights
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Lights


Security System
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Security System


Smart Speaker
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Smart Speaker

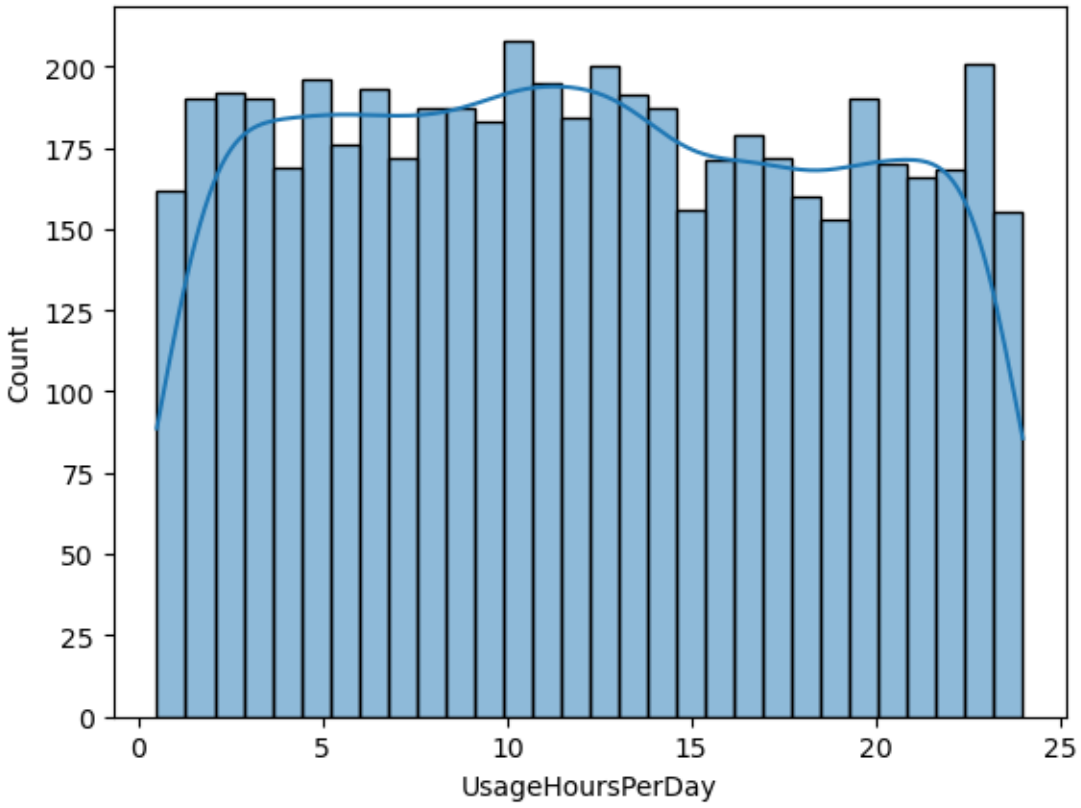


[126]:
```
F.isnull().sum()
```

[126]:
```
UserID                  0
UsageHoursPerDay        0
EnergyConsumption       0
UserPreferences         0
MalfunctionIncidents    0
DeviceAgeMonths         0
SmartHomeEfficiency     0
Camera                  0
Lights                  0
Security System         0
Smart Speaker           0
Thermostat              0
dtype: int64
```

[127]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.histplot(x["UsageHoursPerDay"],bins=30,kde=True)
```

```
plt.show()
```



[128]:
```
F["UsageHoursPerDay"].mean()
```

[128]: 12.052992010466317

[129]:
```
F["UsageHoursPerDay"].median()
```

[129]: 11.903768445051607

[130]:
```
F["UsageHoursPerDay"].mode()[0]
```

[130]: 0.5012414329089748

[131]:
```
F["UsageHoursPerDay"].sum()
```

[131]: 65122.31583254952

[132]:
```
F["UsageHoursPerDay"].min()
```

[132]: 0.5012414329089748

[133]:
```
F["UsageHoursPerDay"].max()
```

[133]: 23.98732600490232

[134]:
```
F["UsageHoursPerDay"].var()
```

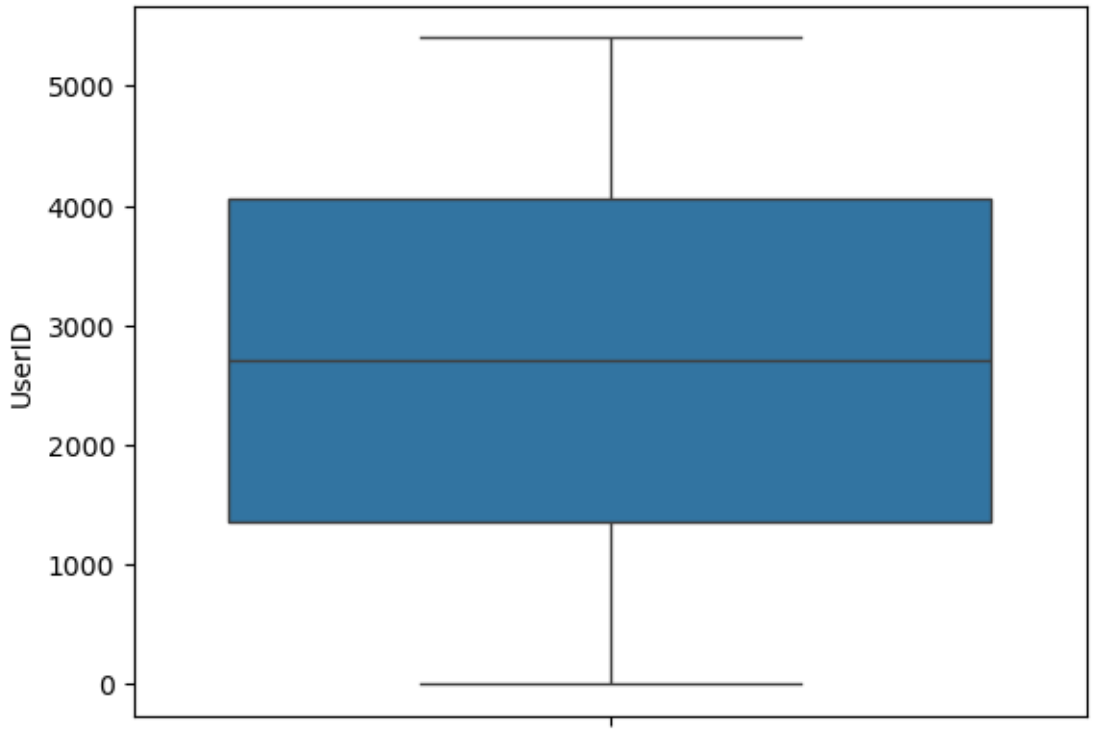
[134]: 45.09069733238142

[135]:
```
F["UsageHoursPerDay"].std()
```

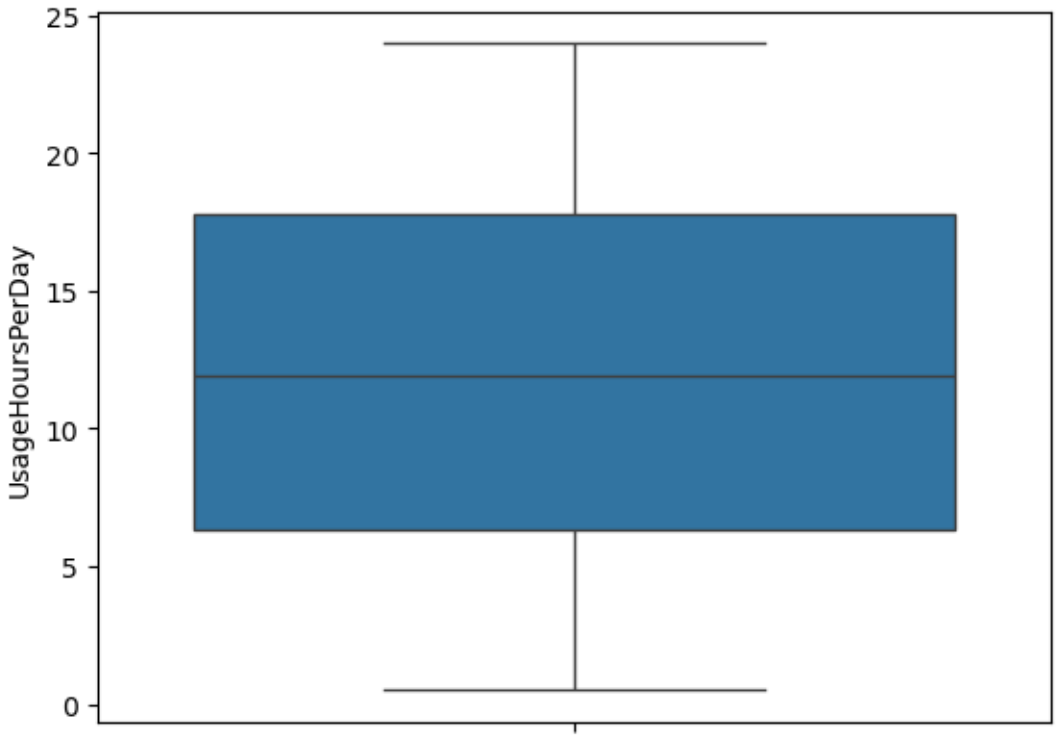
[135]: 6.714960709667736

[136]:
```
import warnings
warnings.filterwarnings("ignore")
for i in F.columns:
    print(i)
    Q1=F[[i]].quantile(0.25)
    Q2=F[[i]].quantile(0.5)
    Q3=F[[i]].quantile(0.75)
    print("Q1=",Q1[0])
    print("Q2=",Q2[0])
    print("Q3=",Q3[0])
    IQR=Q3[0]-Q1[0]
    print("IQR=",IQR)
    upp=Q3[0]+1.5*IQR
    low=Q1[0]-1.5*IQR
    print("LOW=",low)
    print("UPP=",upp)
    sns.boxplot(F[i])
    plt.show()
    s=F.loc[(F[i]>=low)&(F[i]<=upp)]
```

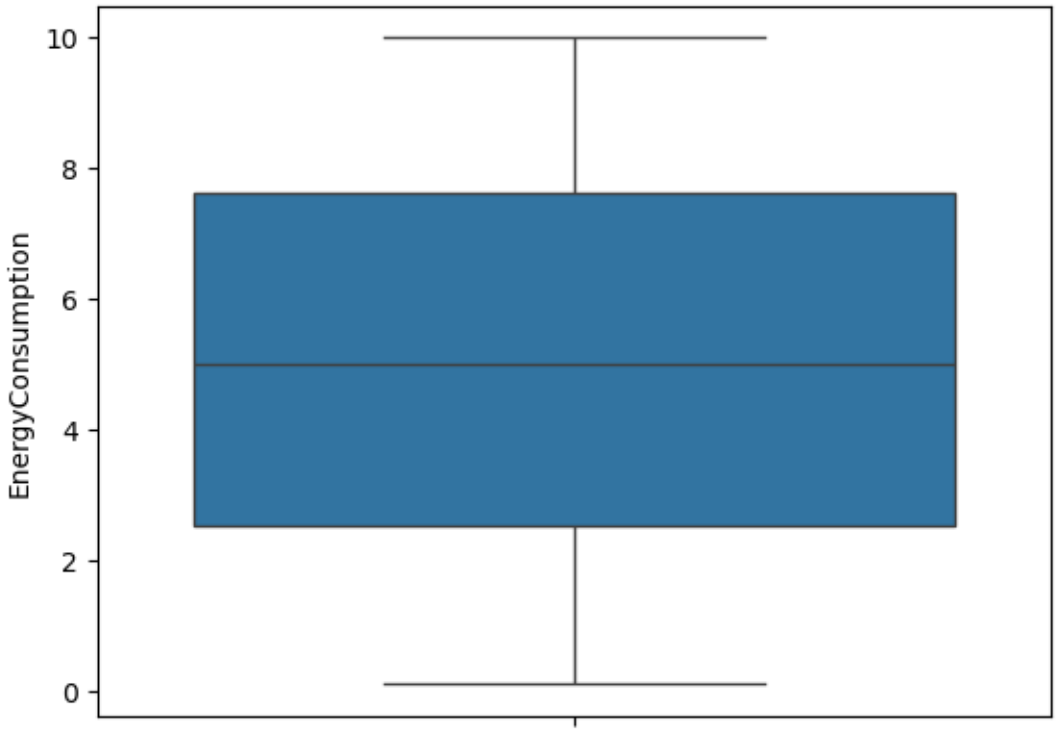
```
UserID
Q1= 1351.5
Q2= 2702.0
Q3= 4052.5
IQR= 2701.0
LOW= -2700.0
UPP= 8104.0
```



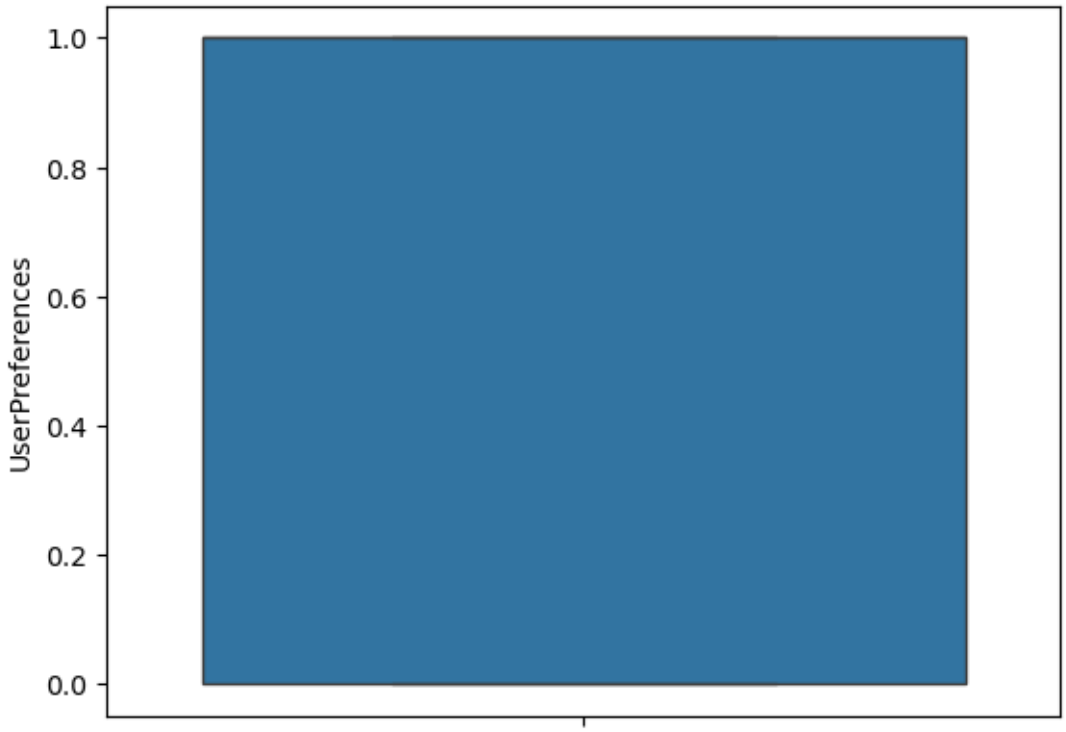

```
UsageHoursPerDay
Q1= 6.297871242333693
Q2= 11.903768445051607
Q3= 17.79175111953669
IQR= 11.493879877202998
LOW= -10.942948573470803
UPP= 35.032570935341184
```



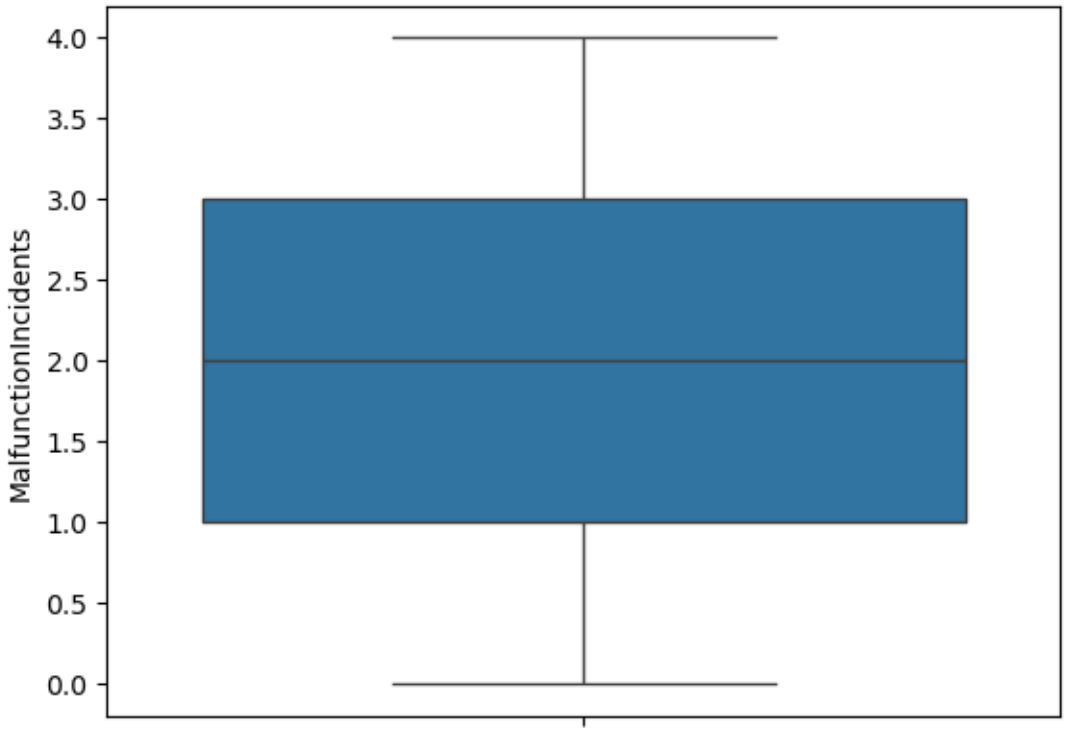

```
EnergyConsumption
Q1= 2.5249681425950814
Q2= 5.007047305947374
Q3= 7.6119115723606425
IQR= 5.086943429765562
LOW= -5.1054470020532605
UPP= 15.242326717008986
```



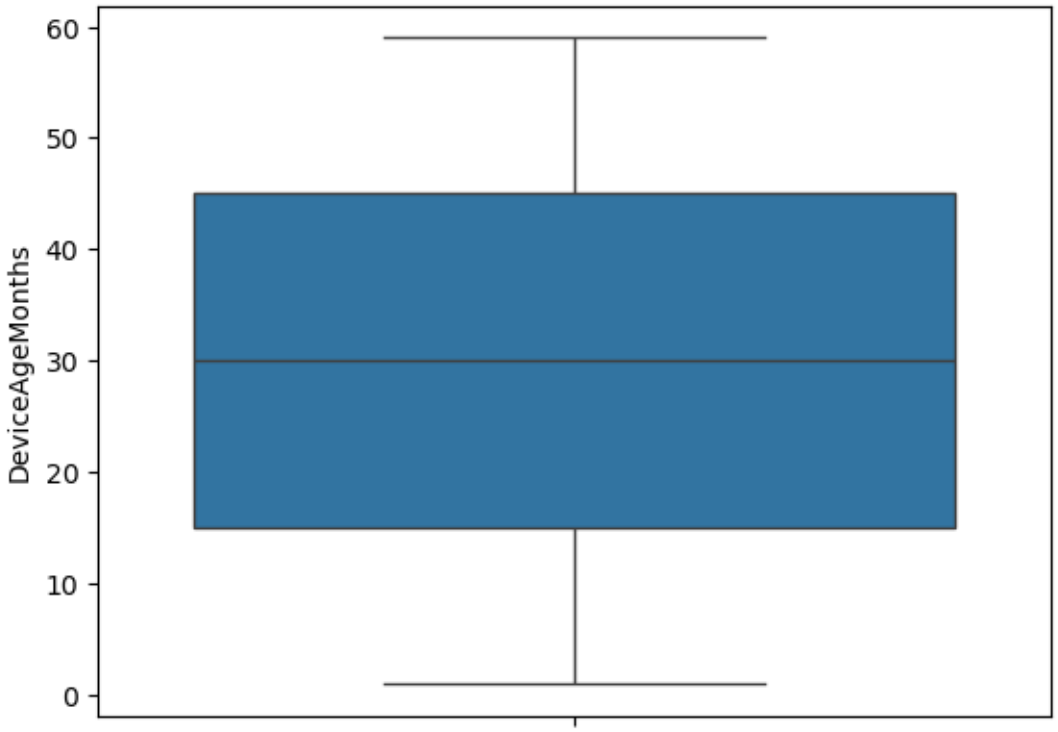

```
UserPreferences
Q1= 0.0
Q2= 1.0
Q3= 1.0
IQR= 1.0
LOW= -1.5
UPP= 2.5
```



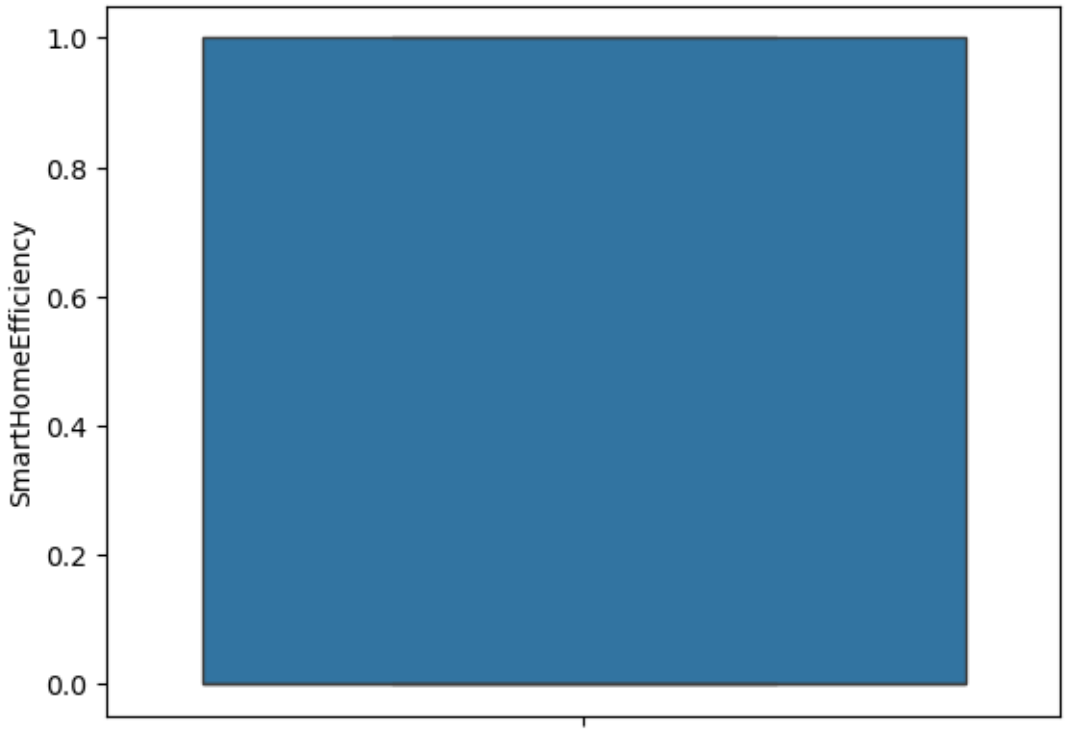

```
MalfunctionIncidents
Q1= 1.0
Q2= 2.0
Q3= 3.0
IQR= 2.0
LOW= -2.0
UPP= 6.0
```



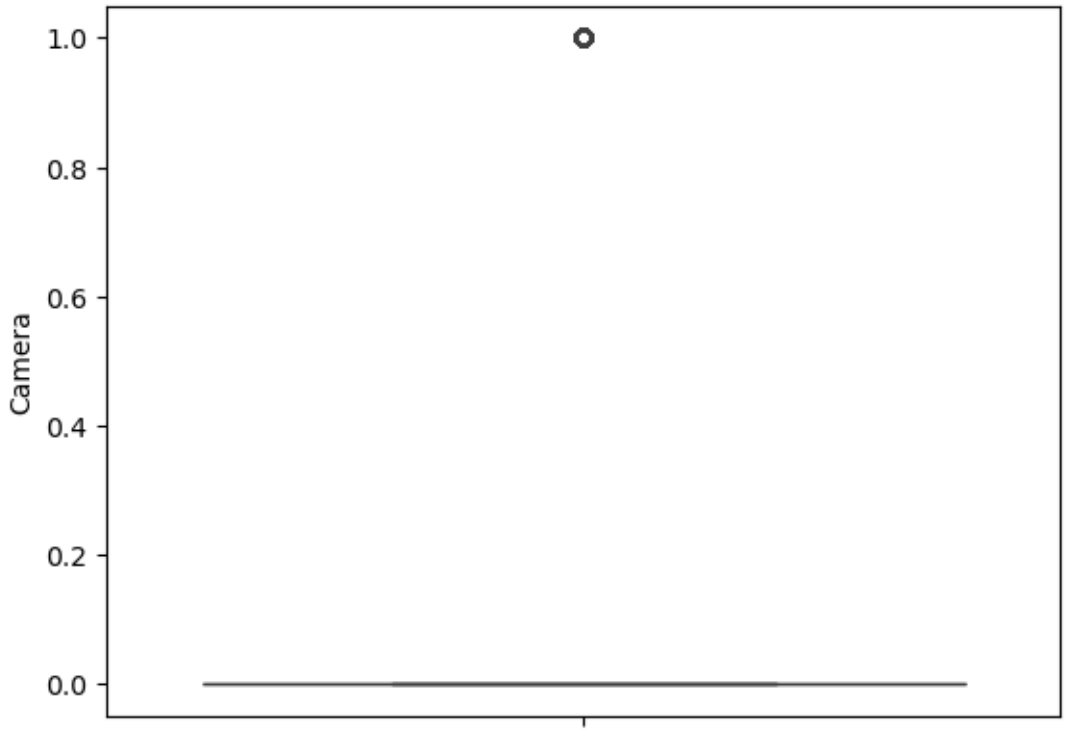

```
DeviceAgeMonths
Q1= 15.0
Q2= 30.0
Q3= 45.0
IQR= 30.0
LOW= -30.0
UPP= 90.0
```



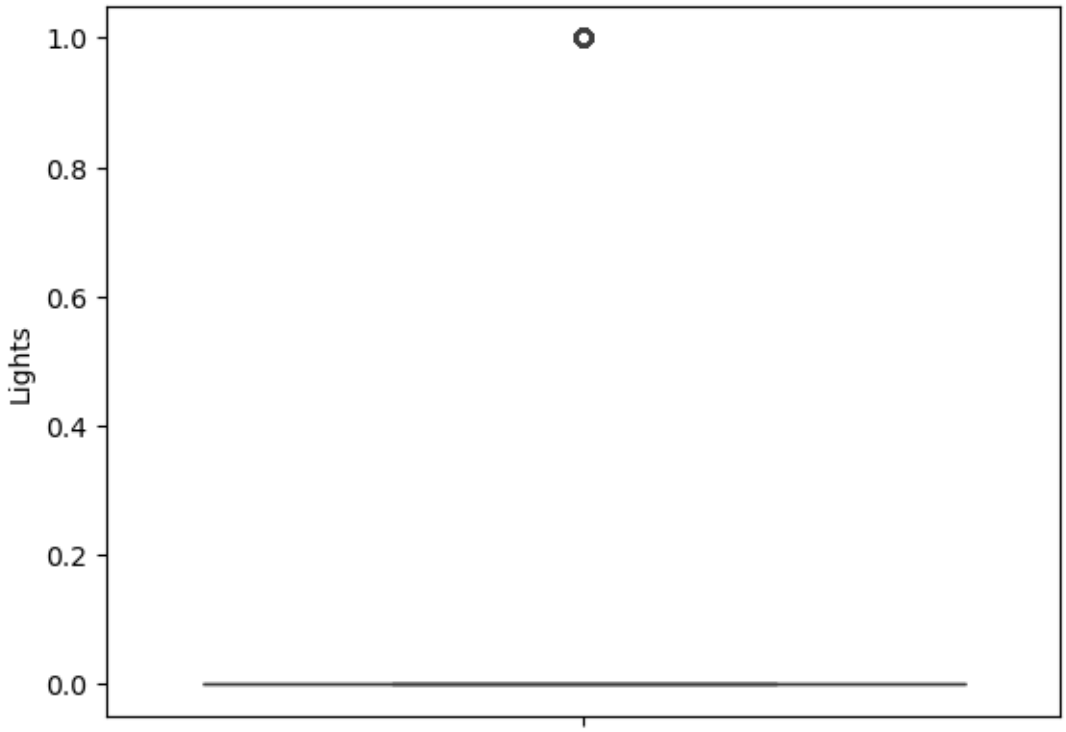

```
SmartHomeEfficiency
Q1= 0.0
Q2= 0.0
Q3= 1.0
IQR= 1.0
LOW= -1.5
UPP= 2.5
```



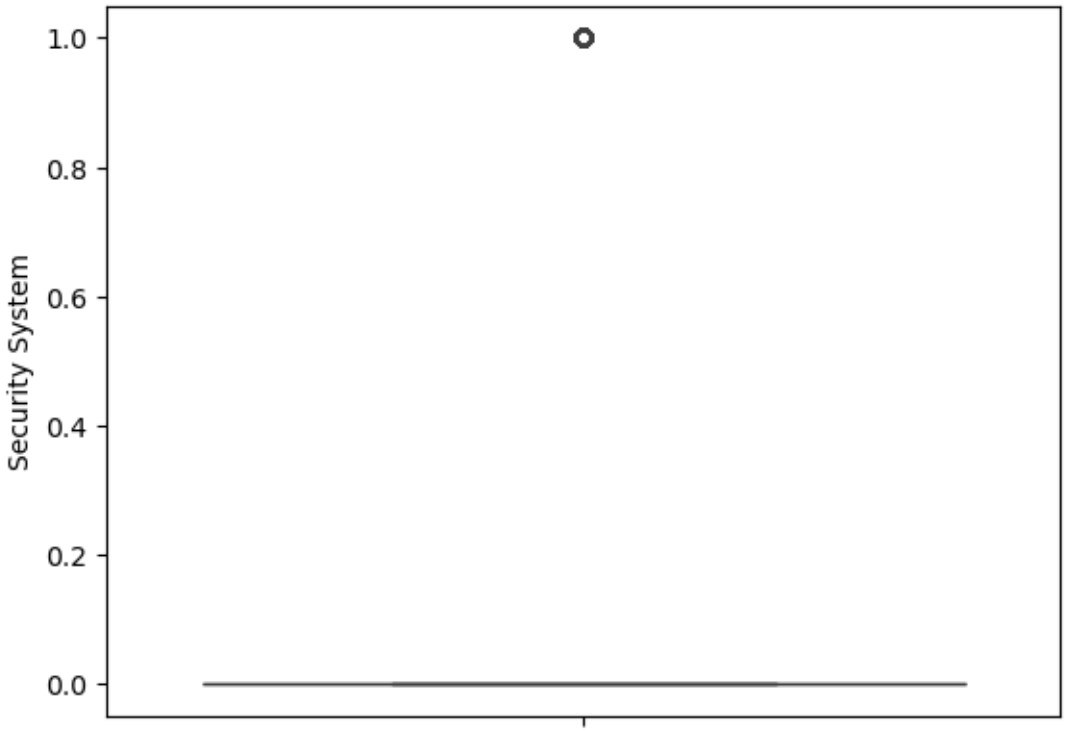

```
Camera
Q1= 0.0
Q2= 0.0
Q3= 0.0
IQR= 0.0
LOW= 0.0
UPP= 0.0
```



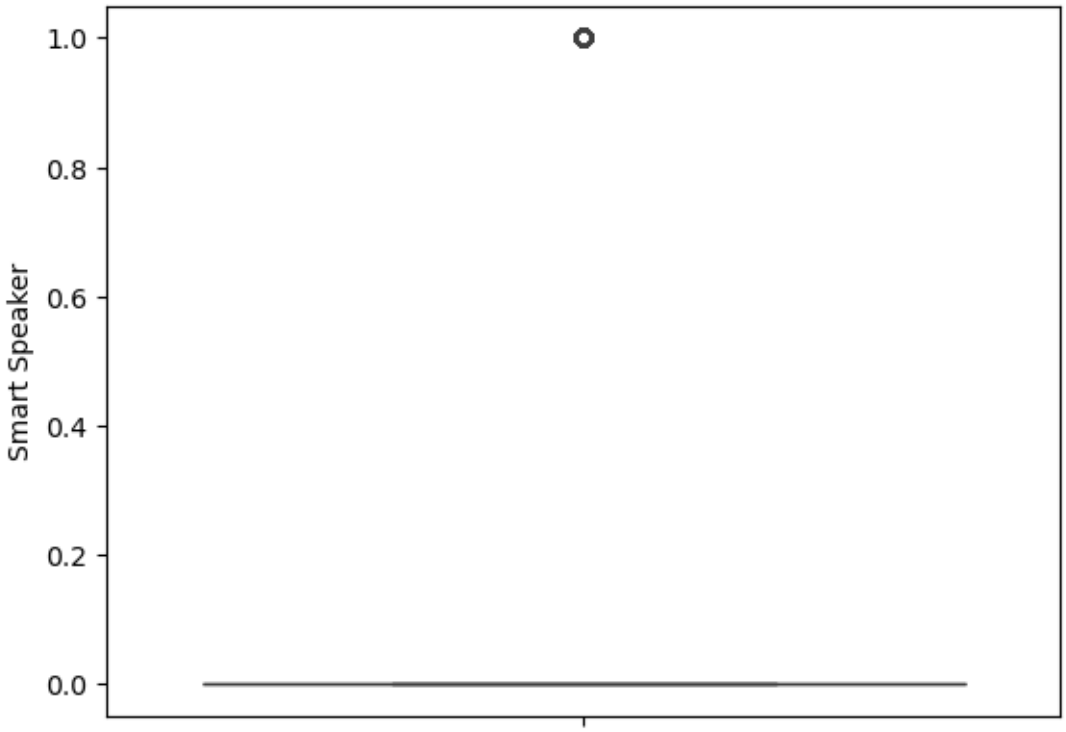

```
Lights
Q1= 0.0
Q2= 0.0
Q3= 0.0
IQR= 0.0
LOW= 0.0
UPP= 0.0
```



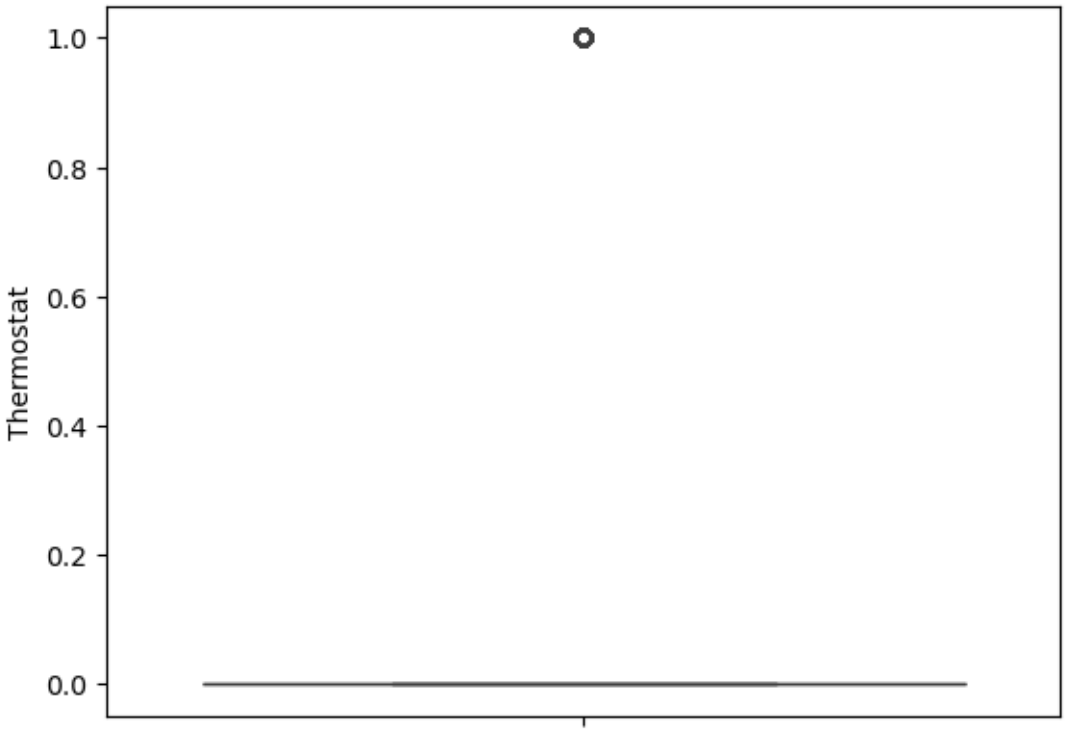

```
Security System
Q1= 0.0
Q2= 0.0
Q3= 0.0
IQR= 0.0
LOW= 0.0
UPP= 0.0
```




```
Smart Speaker
Q1= 0.0
Q2= 0.0
Q3= 0.0
IQR= 0.0
LOW= 0.0
UPP= 0.0
```




```
Thermostat
Q1= 0.0
Q2= 0.0
Q3= 0.0
IQR= 0.0
LOW= 0.0
UPP= 0.0
```

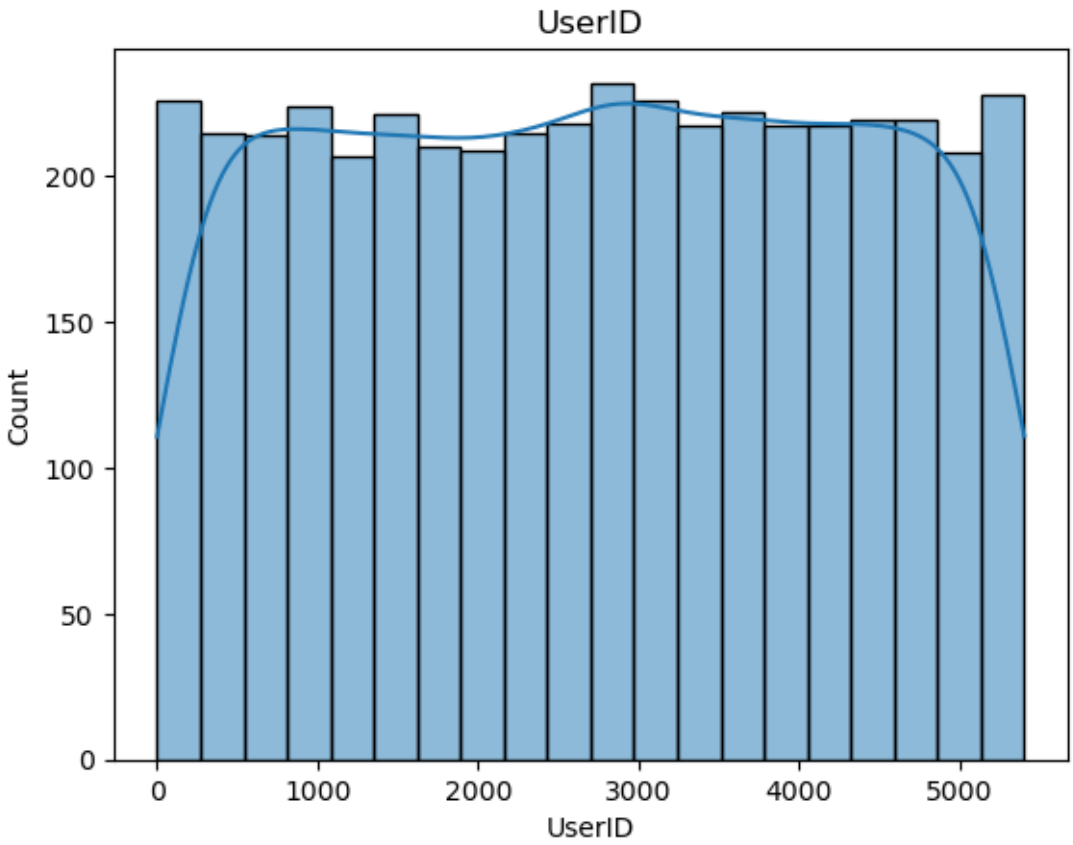


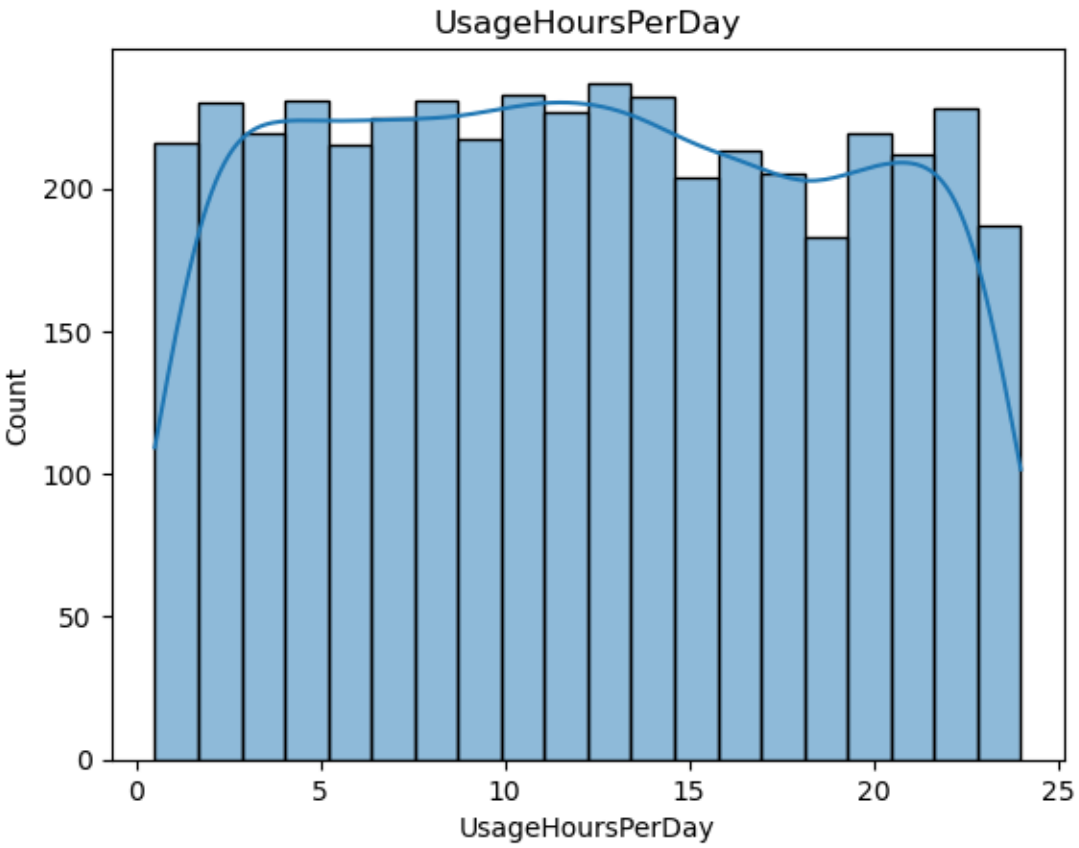
[137]:
```
s
```

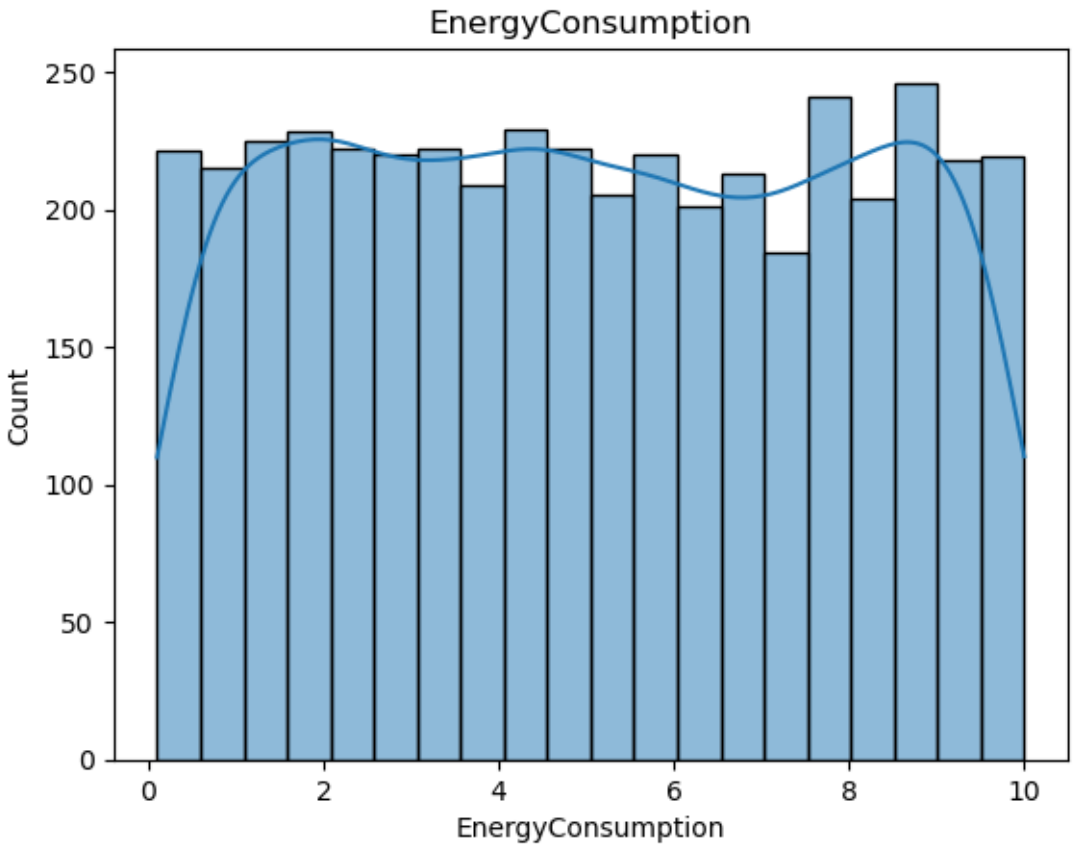
[137]:
```
      UserID  UsageHoursPerDay  EnergyConsumption  UserPreferences  \
0          1         15.307188           1.961607                1
1          2         19.973343           8.610689                1
2          3         18.911535           2.651777                1
3          4          7.011127           2.341653                0
4          5         22.610684           4.859069                1
…        …               …                  …              …
5396    5397         19.301279           0.792446                1
5397    5398          8.633520           4.249140                0
5399    5400          0.561856           1.555992                1
5400    5401         11.096236           7.677779                0
5401    5402          8.782169           7.467929                0

      MalfunctionIncidents  DeviceAgeMonths  SmartHomeEfficiency  Camera  \
0                        4               36                    1       0
1                        0               29                    1       1
2                        0               20                    1       0
3                        3               15                    0       1
4                        3               36                    1       1
…                      …                 …                  …     …
5396                     1               33                    1       1
```

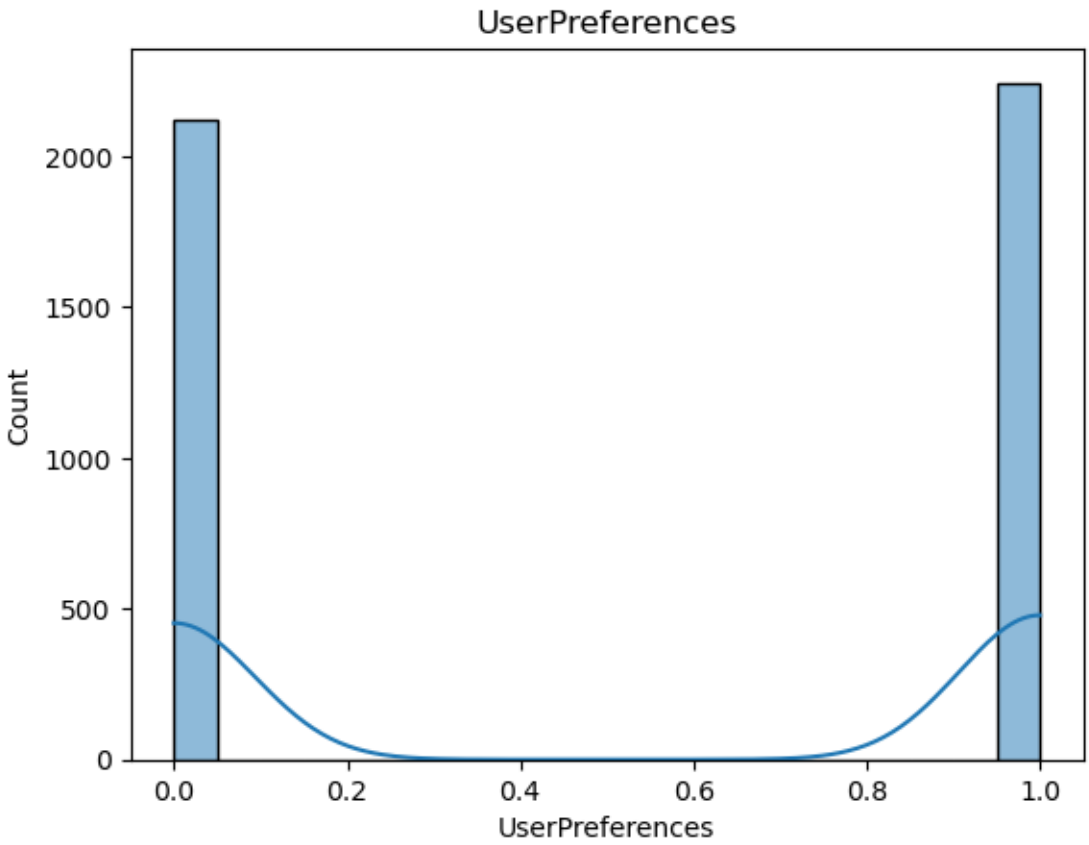
```
5397                    0               32                    0       0
5399                    4               24                    0       0
5400                    0               42                    0       0
5401                    2               28                    1       0

      Lights  Security System  Smart Speaker  Thermostat
0          0                0              1           0
1          0                0              0           0
2          0                1              0           0
3          0                0              0           0
4          0                0              0           0
…      …               …            …          …
5396       0                0              0           0
5397       1                0              0           0
5399       1                0              0           0
5400       0                0              1           0
5401       0                1              0           0

[4364 rows x 12 columns]
```

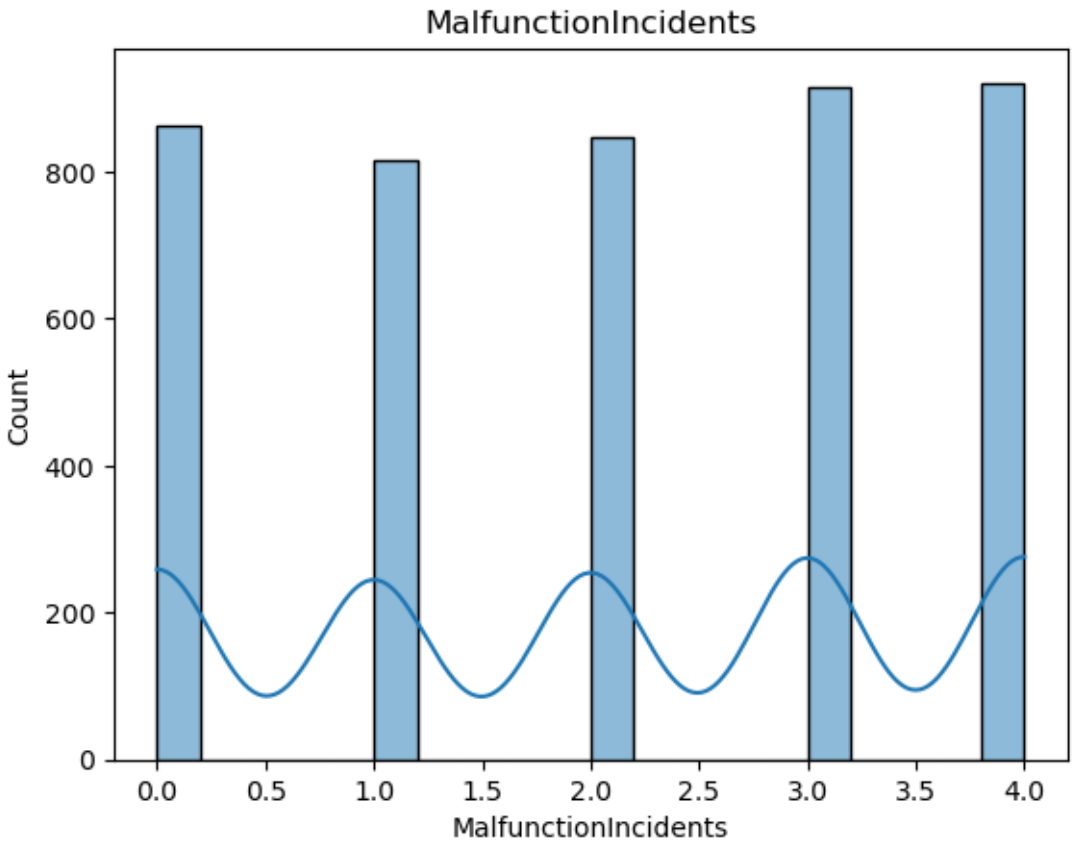
[138]:
```
import matplotlib.pyplot as plt
import seaborn as sns
for i in s.columns:
    sns.histplot(s[i],bins=20,kde=True)
    plt.title(i)
    plt.show()
```

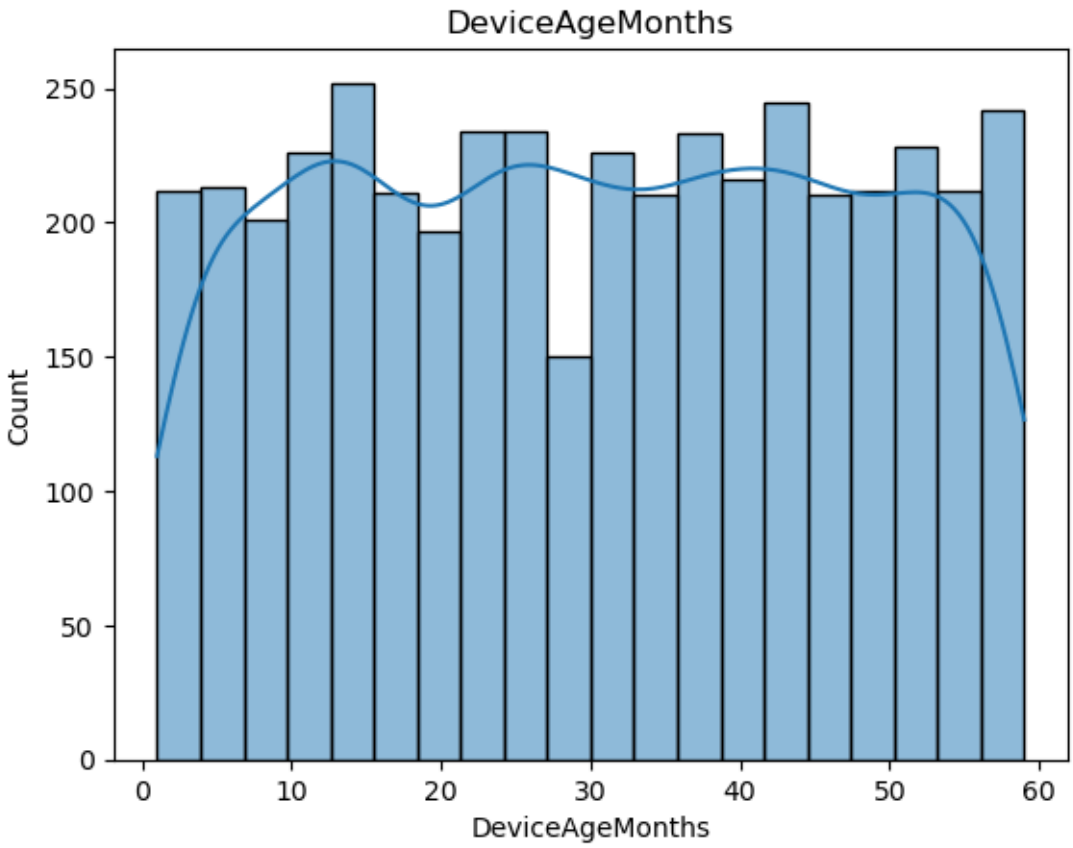

UserID
Count
0
50
100
150
200
0
1000
2000
3000
4000
5000
UserID

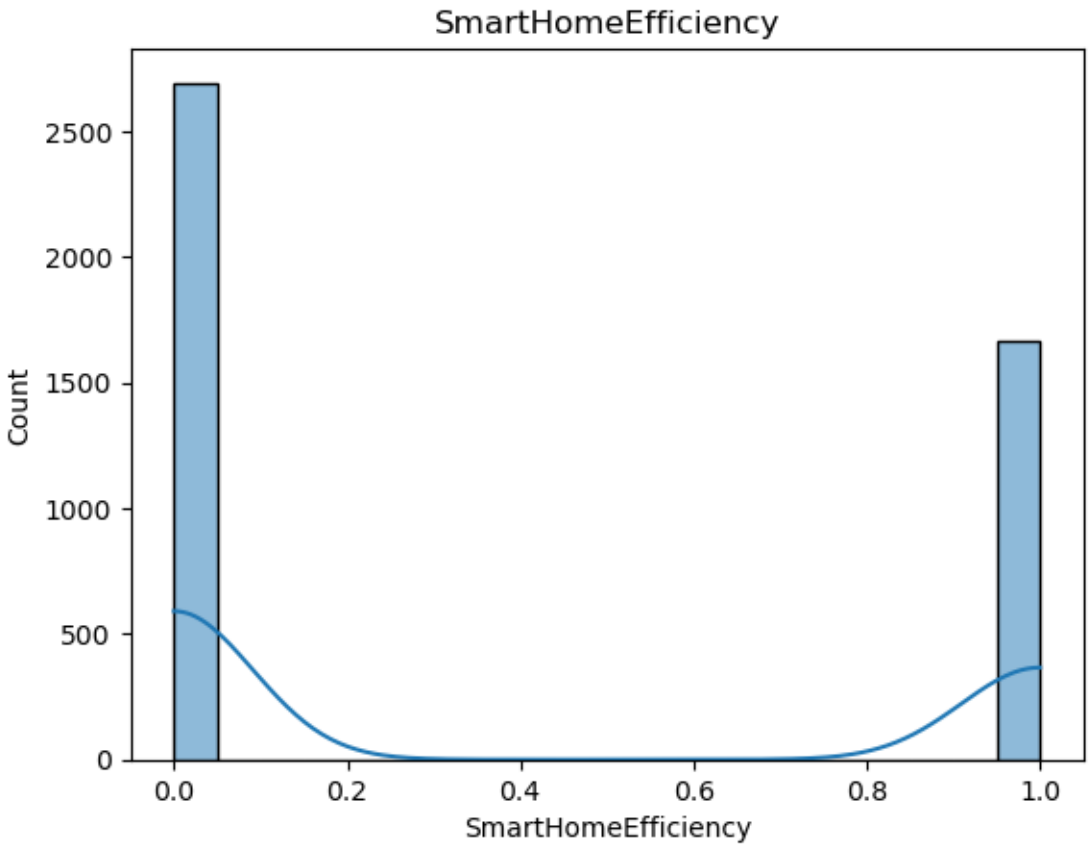

UsageHoursPerDay
Count
0
50
100
150
200
0
5
10
15
20
25
UsageHoursPerDay

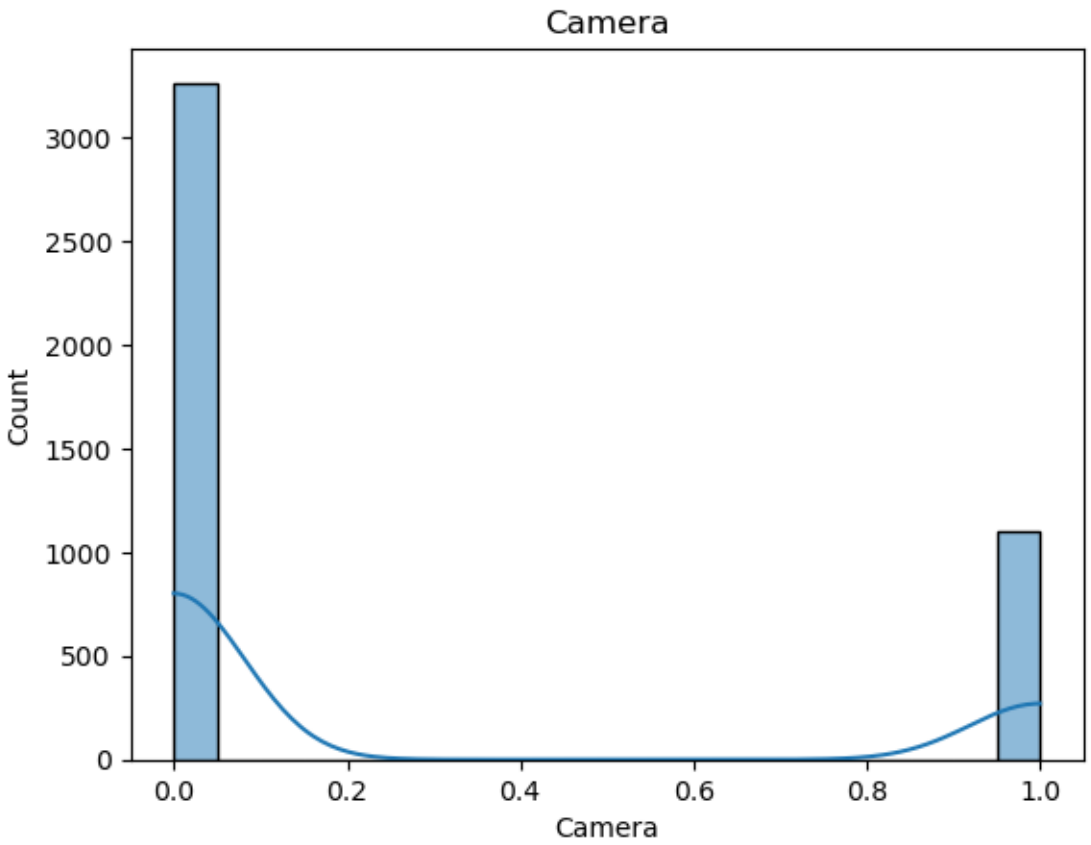

EnergyConsumption
250
200
150
100
50
0
Count
0
2
4
6
8
10
EnergyConsumption

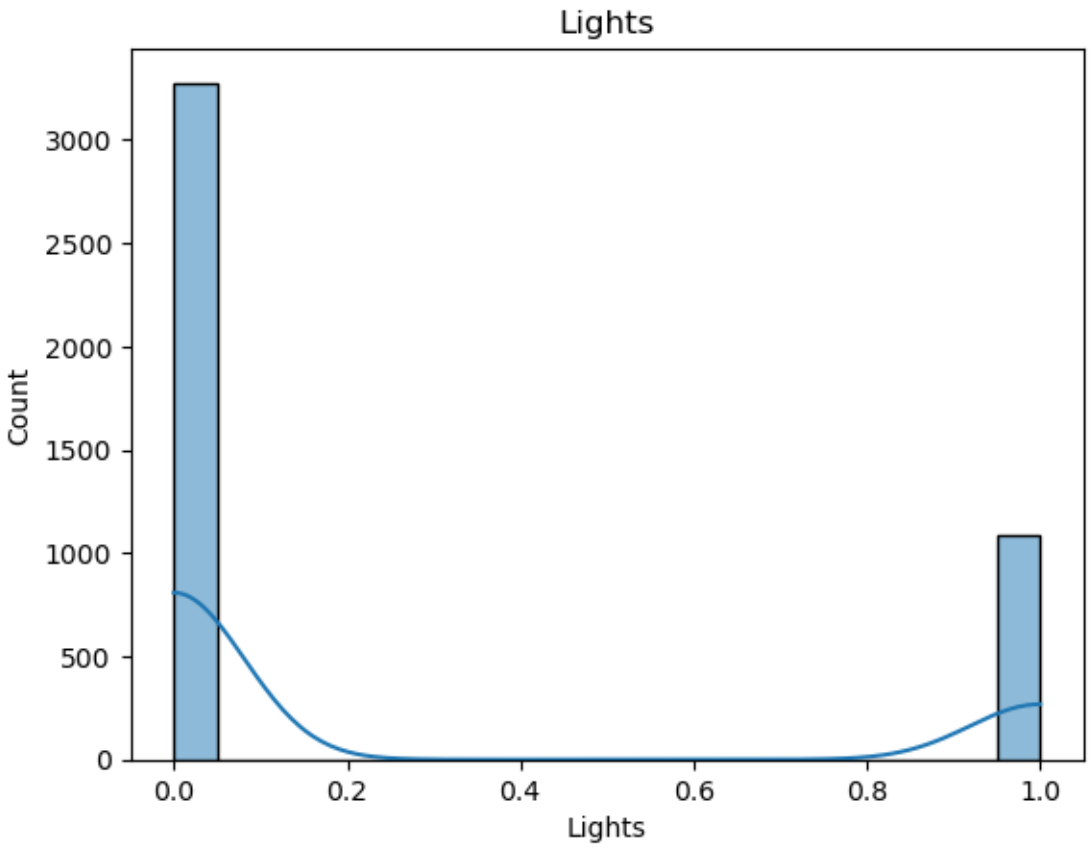

UserPreferences
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
UserPreferences

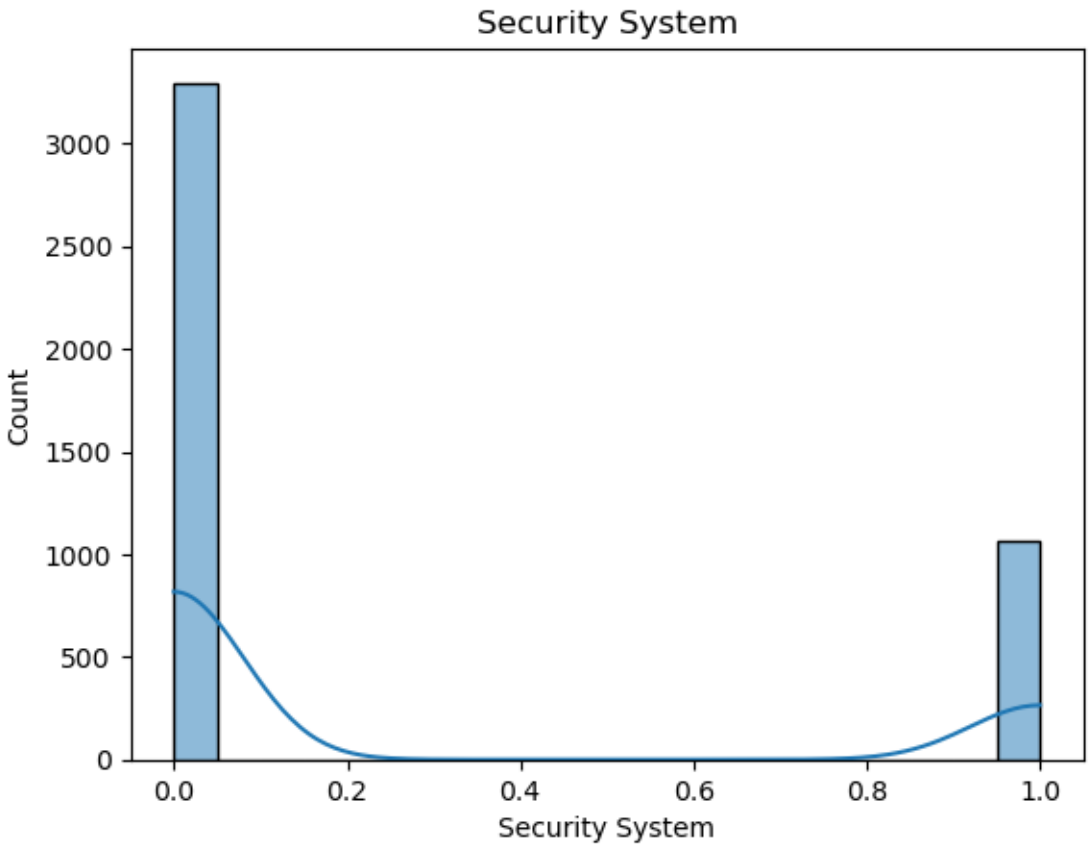

MalfunctionIncidents
Count
0
200
400
600
800
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
MalfunctionIncidents

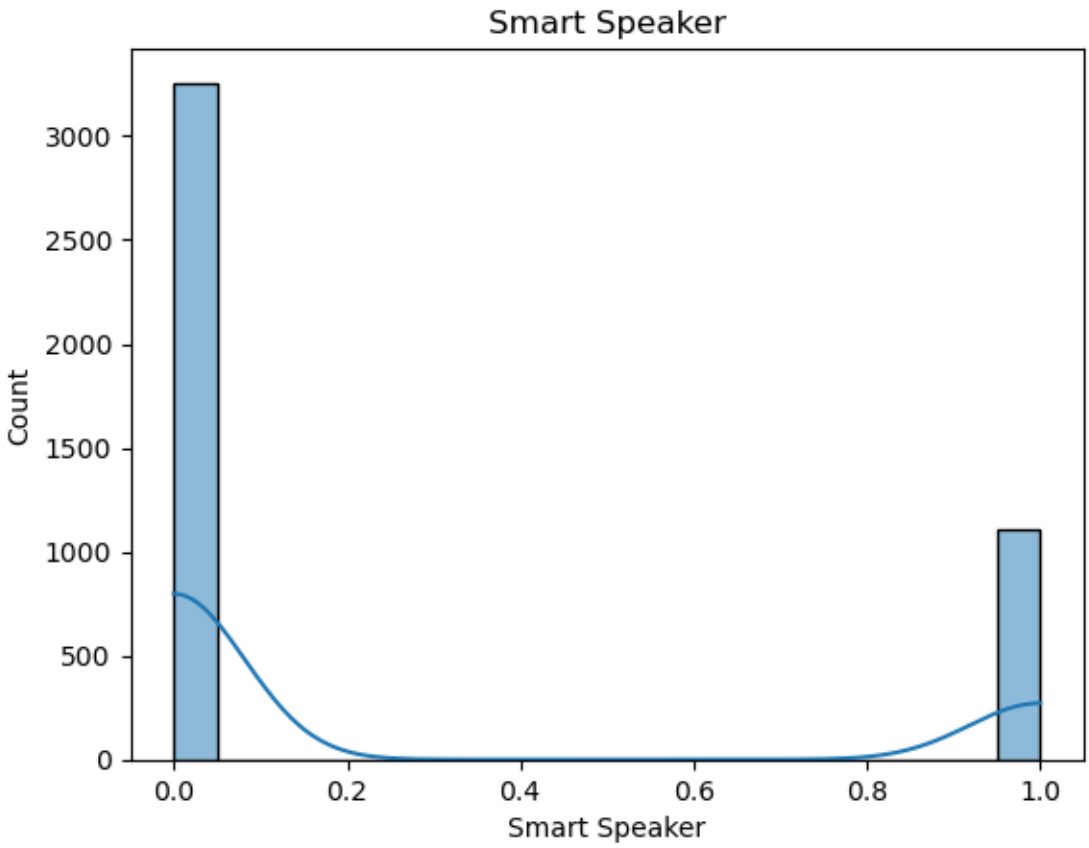

DeviceAgeMonths
Count
DeviceAgeMonths
0
50
100
150
200
250
0
10
20
30
40
50
60

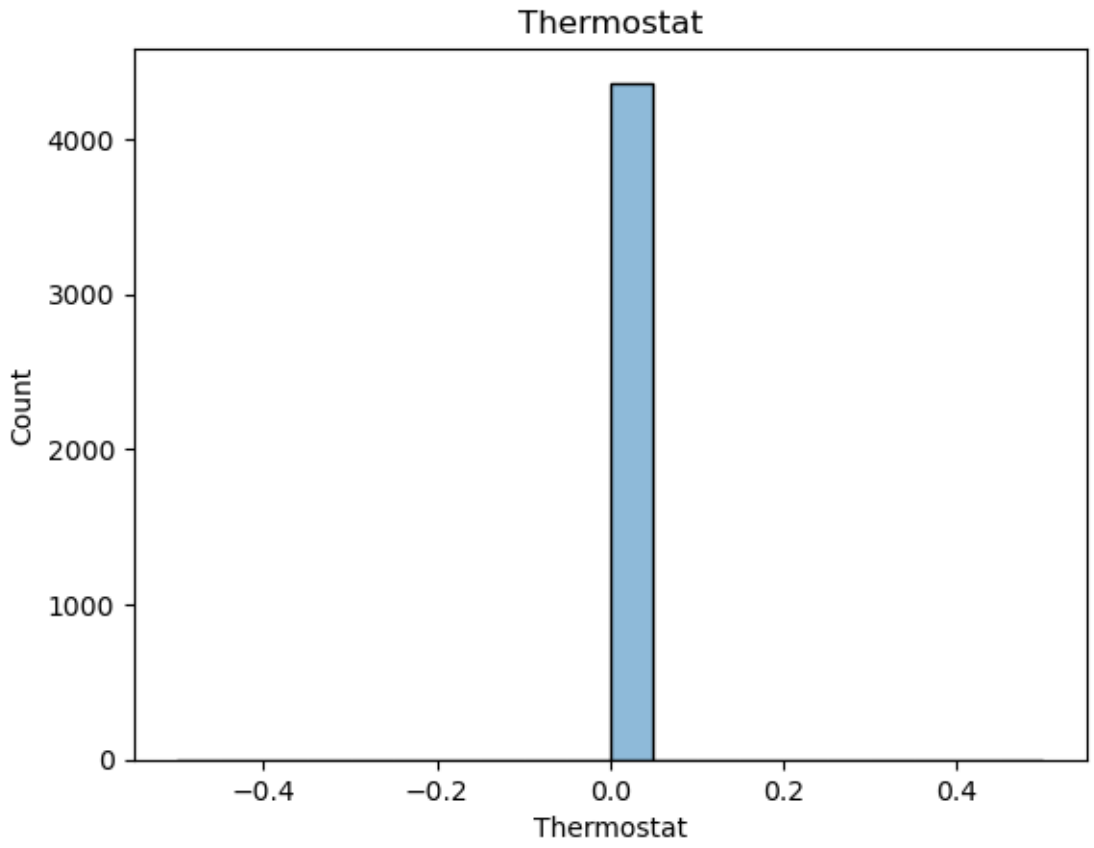

SmartHomeEfficiency
2500
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
SmartHomeEfficiency


Camera
3000
2500
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Camera


Lights
3000
2500
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Lights


Security System
3000
2500
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Security System


Smart Speaker
3000
2500
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Smart Speaker



[139]:
```
for i in s.columns:
    print(i.upper())
    sum=s[i].sum()
    print("sum:",sum)
    mean=s[i].mean()
    print("mean:",mean)
    median=s[i].median()
    print("median:",median)
    mode=s[i].mode()[0]
    print("mode:",mode)
    min=s[i].min()
    print("min:",min)
    max=s[i].max()
    print("max:",max)
    std=s[i].var()
    print("std:",std)
    var=s[i].var()
    print("var:",var)
    print("_"*30)
    plt.show()
```

```
USERID
sum: 11813932
mean: 2707.133822181485
median: 2732.5
mode: 1
min: 1
max: 5402
std: 2433928.7618262344
var: 2433928.7618262344
------------------------------
USAGEHOURSPERDAY
sum: 52544.438568160374
mean: 12.040430469330975
median: 11.915358879084705
mode: 0.5053353981290052
min: 0.5053353981290052
max: 23.98732600490232
std: 45.140528007364715
var: 45.140528007364715
------------------------------
ENERGYCONSUMPTION
sum: 21963.40730917837
mean: 5.0328614365669955
median: 4.968829676983313
mode: 0.1015616713227616
min: 0.1015616713227616
max: 9.998070926924637
std: 8.266604441984045
var: 8.266604441984045
------------------------------
USERPREFERENCES
sum: 2243
mean: 0.5139780018331805
median: 1.0
mode: 1
min: 0
max: 1
std: 0.24986187070551405
var: 0.24986187070551405
------------------------------
MALFUNCTIONINCIDENTS
sum: 8941
mean: 2.048808432630614
median: 2.0
mode: 4
min: 0
max: 4
std: 2.030851204182858
```

var: 2.030851204182858

DEVICEAGEMONTHS
sum: 132041
mean: 30.256874427131073
median: 30.0
mode: 13
min: 1
max: 59
std: 288.16526329754504
var: 288.16526329754504

SMARTHOMEEFFICIENCY
sum: 1670
mean: 0.38267644362969755
median: 0.0
mode: 0
min: 0
max: 1
std: 0.23628932824624083
var: 0.23628932824624083

CAMERA
sum: 1101
mean: 0.2522914757103575
median: 0.0
mode: 0
min: 0
max: 1
std: 0.18868372341115514
var: 0.18868372341115514

LIGHTS
sum: 1087
mean: 0.24908340971585702
median: 0.0
mode: 0
min: 0
max: 1
std: 0.1870837345035399
var: 0.1870837345035399

SECURITY SYSTEM
sum: 1068
mean: 0.2447296058661778
median: 0.0
mode: 0
min: 0

```
max: 1
std: 0.18487939054203606
var: 0.18487939054203606
------------------------------
SMART SPEAKER
sum: 1108
mean: 0.2538955087076077
median: 0.0
mode: 0
min: 0
max: 1
std: 0.18947599733025922
var: 0.18947599733025922
------------------------------
THERMOSTAT
sum: 0
mean: 0.0
median: 0.0
mode: 0
min: 0
max: 0
std: 0.0
var: 0.0
------------------------------
```

[140]:
```
s["Smart Speaker"].nunique()
```

[140]:
```
2
```

[141]:
```
F["Smart Speaker"].value_counts()
```

[141]:
```
Smart Speaker
0    4295
1    1108
Name: count, dtype: int64
```

[142]:
```
s["Smart Speaker"].value_counts()
```

[142]:
```
Smart Speaker
0    3256
1    1108
Name: count, dtype: int64
```

[143]:
```
F["SmartHomeEfficiency"].value_counts()
```

[143]:
```
SmartHomeEfficiency
0    3368
```

```
1    2035
Name: count, dtype: int64
```

[144]:
```
s["SmartHomeEfficiency"].value_counts()
```

[144]:
```
SmartHomeEfficiency
0    2694
1    1670
Name: count, dtype: int64
```

[145]:
```
s.corr()
```

[145]:

	UserID	UsageHoursPerDay	EnergyConsumption	\
UserID	1.000000	-0.004086	-0.012594	
UsageHoursPerDay	-0.004086	1.000000	0.026804	
EnergyConsumption	-0.012594	0.026804	1.000000	
UserPreferences	0.022984	-0.000396	0.013514	
MalfunctionIncidents	-0.007615	0.004248	-0.010932	
DeviceAgeMonths	0.004980	0.017997	-0.011321	
SmartHomeEfficiency	0.047688	0.188698	-0.179222	
Camera	-0.004935	0.006312	0.009659	
Lights	-0.022023	0.001047	0.002409	
Security System	0.022982	-0.002057	0.020856	
Smart Speaker	0.004107	-0.005308	-0.032634	
Thermostat	NaN	NaN	NaN	

	UserPreferences	MalfunctionIncidents	DeviceAgeMonths	\
UserID	0.022984	-0.007615	0.004980	
UsageHoursPerDay	-0.000396	0.004248	0.017997	
EnergyConsumption	0.013514	-0.010932	-0.011321	
UserPreferences	1.000000	-0.007232	-0.024126	
MalfunctionIncidents	-0.007232	1.000000	-0.023807	
DeviceAgeMonths	-0.024126	-0.023807	1.000000	
SmartHomeEfficiency	0.604322	-0.121928	-0.199738	
Camera	0.005394	0.015648	-0.007579	
Lights	0.000324	-0.011176	-0.011682	
Security System	-0.011654	-0.000048	0.007683	
Smart Speaker	0.005807	-0.004463	0.011582	
Thermostat	NaN	NaN	NaN	

	SmartHomeEfficiency	Camera	Lights	\
UserID	0.047688	-0.004935	-0.022023	
UsageHoursPerDay	0.188698	0.006312	0.001047	
EnergyConsumption	-0.179222	0.009659	0.002409	
UserPreferences	0.604322	0.005394	0.000324	
MalfunctionIncidents	-0.121928	0.015648	-0.011176	
DeviceAgeMonths	-0.199738	-0.007579	-0.011682	

```
SmartHomeEfficiency              1.000000  0.005073 -0.003237
Camera                           0.005073  1.000000 -0.334550
Lights                          -0.003237 -0.334550  1.000000
Security System                 -0.008442 -0.330657 -0.327845
Smart Speaker                    0.006493 -0.338854 -0.335973
Thermostat                            NaN       NaN       NaN

                     Security System  Smart Speaker  Thermostat
UserID                      0.022982       0.004107         NaN
UsageHoursPerDay           -0.002057      -0.005308         NaN
EnergyConsumption           0.020856      -0.032634         NaN
UserPreferences            -0.011654       0.005807         NaN
MalfunctionIncidents       -0.000048      -0.004463         NaN
DeviceAgeMonths             0.007683       0.011582         NaN
SmartHomeEfficiency        -0.008442       0.006493         NaN
Camera                     -0.330657      -0.338854         NaN
Lights                     -0.327845      -0.335973         NaN
Security System             1.000000      -0.332062         NaN
Smart Speaker              -0.332062       1.000000         NaN
Thermostat                       NaN            NaN         NaN
```

[146]:
```
F=s.drop("SmartHomeEfficiency",axis=1)
T=s["SmartHomeEfficiency"]
```

[147]:
```
F.head()
```

[147]:
```
   UserID  UsageHoursPerDay  EnergyConsumption  UserPreferences  \
0       1         15.307188           1.961607                1
1       2         19.973343           8.610689                1
2       3         18.911535           2.651777                1
3       4          7.011127           2.341653                0
4       5         22.610684           4.859069                1

   MalfunctionIncidents  DeviceAgeMonths  Camera  Lights  Security System  \
0                     4               36       0       0                0
1                     0               29       1       0                0
2                     0               20       0       0                1
3                     3               15       1       0                0
4                     3               36       1       0                0

   Smart Speaker  Thermostat
0              1           0
1              0           0
2              0           0
3              0           0
4              0           0
```

[148]:
```
T
```

[148]:
```
0       1
1       1
2       1
3       0
4       1
       ..
5396    1
5397    0
5399    0
5400    0
5401    1
Name: SmartHomeEfficiency, Length: 4364, dtype: int64
```

[149]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(F,T,train_size=0.80)
```

[150]:
```
X_train.shape
```

[150]:
```
(3491, 11)
```

[151]:
```
y_train.shape
```

[151]:
```
(3491,)
```

[152]:
```
T.unique()
```

[152]:
```
array([1, 0], dtype=int64)
```

[153]:
```
import warnings
warnings.filterwarnings("ignore")
from sklearn.linear_model import LogisticRegression
L=LogisticRegression()
from sklearn.model_selection import GridSearchCV
params={"C":[0.1,0.01,0.2,0.04,0.001],"penalty":["l1","l2"]}
log=GridSearchCV(L,param_grid=params,scoring="accuracy",cv=5)
```

[154]:
```
log.fit(X_train,y_train)
```

[154]:
```
GridSearchCV(cv=5, estimator=LogisticRegression(),
             param_grid={'C': [0.1, 0.01, 0.2, 0.04, 0.001],
                         'penalty': ['l1', 'l2']},
             scoring='accuracy')
```

[155]:
```
log.best_params_
```

[155]:
```
{'C': 0.04, 'penalty': 'l2'}
```

[156]:
```
model=log.best_estimator_
model
```

[156]:
```
LogisticRegression(C=0.04)
```

[157]:
```
model.score(X_train,y_train)
```

[157]:
```
0.8627900315095961
```

[158]:
```
model.score(X_test,y_test)
```

[158]:
```
0.861397479954181
```

[159]:
```
from sklearn.neighbors import KNeighborsClassifier
A=KNeighborsClassifier()
from sklearn.model_selection import GridSearchCV
knn=GridSearchCV(A,param_grid={"n_neighbors":[3,5,7,9]},scoring="accuracy",cv=5)
```

[160]:
```
knn.fit(X_train,y_train)
```

[160]:
```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': [3, 5, 7, 9]}, scoring='accuracy')
```

[161]:
```
knn.best_params_
```

[161]:
```
{'n_neighbors': 5}
```

[162]:
```
model2=knn.best_estimator_
model2
```

[162]:
```
KNeighborsClassifier()
```

[163]:
```
model2.score(X_train,y_train)
```

[163]:
```
0.7419077628186765
```

[164]:
```
model2.score(X_test,y_test)
```

[164]:
```
0.5807560137457045
```

[165]:
```
from sklearn.naive_bayes import␣
 ↪GaussianNB,MultinomialNB,BernoulliNB,ComplementNB
Gaussian=GaussianNB()
Multinomial=MultinomialNB()
Bernoulli=BernoulliNB()
```

```
Complement=ComplementNB()
```

[166]:
```
Gaussian.fit(X_train,y_train)
```

[166]:
```
GaussianNB()
```

[167]:
```
Gaussian.score(X_train,y_train)
```

[167]:
```
0.8341449441420796
```

[168]:
```
Gaussian.score(X_test,y_test)
```

[168]:
```
0.8384879725085911
```

[169]:
```
Multinomial.fit(X_train,y_train)
```

[169]:
```
MultinomialNB()
```

[170]:
```
Multinomial.score(X_train,y_train)
```

[170]:
```
0.6264680607275852
```

[171]:
```
Multinomial.score(X_test,y_test)
```

[171]:
```
0.6185567010309279
```

[172]:
```
Bernoulli.fit(X_train,y_train)
```

[172]:
```
BernoulliNB()
```

[173]:
```
Bernoulli.score(X_train,y_train)
```

[173]:
```
0.791177313090805
```

[174]:
```
Bernoulli.score(X_test,y_test)
```

[174]:
```
0.7869415807560137
```

[175]:
```
Complement.fit(X_train,y_train)
```

[175]:
```
ComplementNB()
```

[176]:
```
Complement.score(X_train,y_train)
```

[176]:
```
0.6218848467487826
```

[177]:
```
Complement.score(X_test,y_test)
```

[177]:
```
0.6071019473081328
```

[178]:
```
from sklearn.naive_bayes import GaussianNB,BernoulliNB,ComplementNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

[179]:
```
models={"Gaussian":GaussianNB(),
        "Bernoulli":BernoulliNB(),
        "Complement":ComplementNB(),
        "SVC":SVC(),
        "KNN":KNeighborsClassifier(),
        "Logistic":LogisticRegression(),
       }
```

[180]:
```
models.keys()
```

[180]:
```
dict_keys(['Gaussian', 'Bernoulli', 'Complement', 'SVC', 'KNN', 'Logistic'])
```

[181]:
```
models.values()
```

[181]:
```
dict_values([GaussianNB(), BernoulliNB(), ComplementNB(), SVC(),
KNeighborsClassifier(), LogisticRegression()])
```

[182]:
```
models.items()
```

[182]:
```
dict_items([('Gaussian', GaussianNB()), ('Bernoulli', BernoulliNB()),
('Complement', ComplementNB()), ('SVC', SVC()), ('KNN', KNeighborsClassifier()),
('Logistic', LogisticRegression())])
```

[183]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

[184]:
```
Data=[]
for i,j in models.items():
    j.fit(X_train,y_train)
    TR=j.score(X_train,y_train)
    TE=j.score(X_test,y_test)
    pred=j.predict(X_test)
    Data.append([i,TR,TE])
    print(i.capitalize())
    print(classification_report(y_test,pred))
    print(confusion_matrix(y_test,pred))
    print("_"*80)
```

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

```
              0       0.93      0.79      0.86       536
              1       0.74      0.91      0.81       337

    accuracy                            0.84       873
   macro avg       0.83      0.85      0.84       873
weighted avg       0.86      0.84      0.84       873

[[426 110]
 [ 31 306]]
--------------------------------------------------------------------------------
Bernoulli
              precision    recall  f1-score   support

              0       0.93      0.71      0.80       536
              1       0.66      0.91      0.77       337

    accuracy                            0.79       873
   macro avg       0.80      0.81      0.79       873
weighted avg       0.83      0.79      0.79       873

[[379 157]
 [ 29 308]]
--------------------------------------------------------------------------------
Complement
              precision    recall  f1-score   support

              0       0.71      0.61      0.66       536
              1       0.49      0.60      0.54       337

    accuracy                            0.61       873
   macro avg       0.60      0.61      0.60       873
weighted avg       0.63      0.61      0.61       873

[[328 208]
 [135 202]]
--------------------------------------------------------------------------------
Svc
              precision    recall  f1-score   support

              0       0.61      1.00      0.76       536
              1       0.00      0.00      0.00       337

    accuracy                            0.61       873
   macro avg       0.31      0.50      0.38       873
weighted avg       0.38      0.61      0.47       873

[[536   0]
 [337   0]]
```

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

           0       0.65      0.69      0.67       536
           1       0.45      0.40      0.42       337

    accuracy                           0.58       873
   macro avg       0.55      0.55      0.55       873
weighted avg       0.57      0.58      0.58       873

[[372 164]
 [202 135]]
--------------------------------------------------------------------------------
Logistic
              precision    recall  f1-score   support

           0       0.90      0.88      0.89       536
           1       0.81      0.85      0.83       337

    accuracy                           0.86       873
   macro avg       0.86      0.86      0.86       873
weighted avg       0.87      0.86      0.87       873

[[469  67]
 [ 51 286]]
--------------------------------------------------------------------------------
```

[185]:
```
Data
```

[185]:
```
[['Gaussian', 0.8341449441420796, 0.8384879725085911],
 ['Bernoulli', 0.791177313090805, 0.7869415807560137],
 ['Complement', 0.6218848467487826, 0.6071019473081328],
 ['SVC', 0.6181609853910054, 0.6139747995418099],
 ['KNN', 0.7419077628186765, 0.5807560137457045],
 ['Logistic', 0.8645087367516471, 0.8648339060710195]]
```

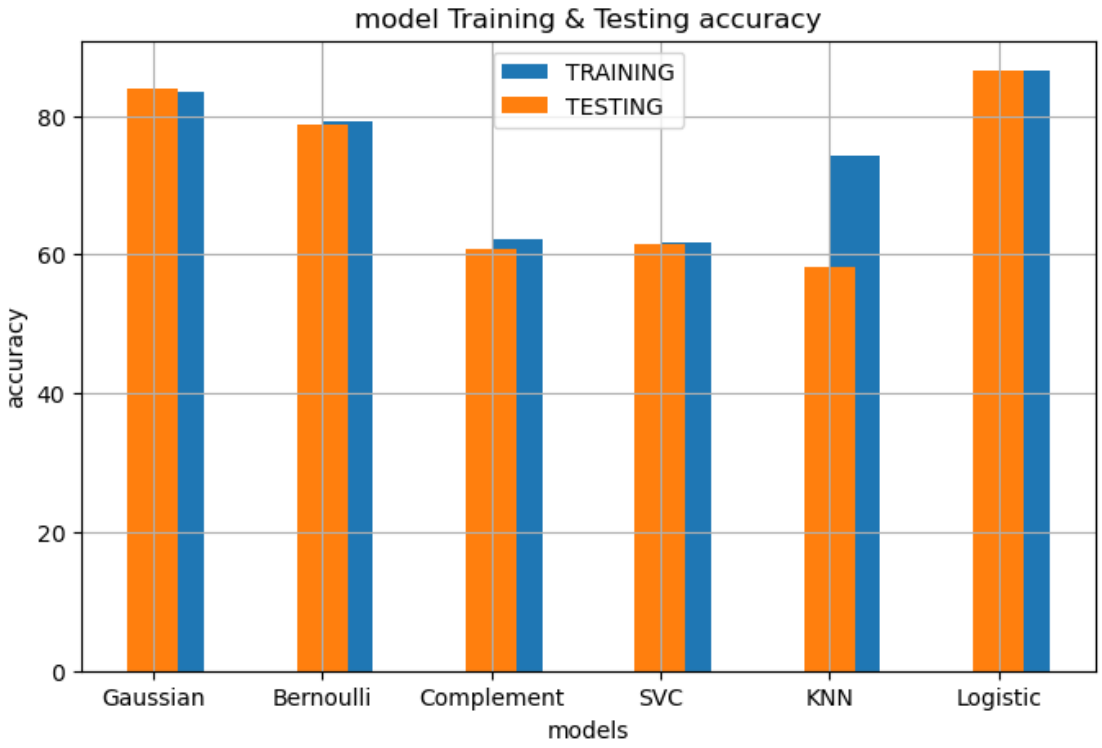
[186]:
```
Result=pd.DataFrame(Data,columns=("models name","Training Accuracy","Testing␣
↪Accuracy"))
Result
```

[186]:

	models name	Training Accuracy	Testing Accuracy
0	Gaussian	0.834145	0.838488
1	Bernoulli	0.791177	0.786942
2	Complement	0.621885	0.607102
3	SVC	0.618161	0.613975
4	KNN	0.741908	0.580756

```
5     Logistic            0.864509           0.864834
```

[187]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,5))
plt.bar(Result["models name"],Result["Training Accuracy"]*100,width=0.
  ↪3,align="edge",label="TRAINING")
plt.bar(Result["models name"],Result["Testing Accuracy"]*100,width=0.
  ↪3,align="center",label="TESTING")
plt.grid()
plt.legend()
plt.xlabel("models")
plt.ylabel("accuracy")
plt.title("model Training & Testing accuracy")
plt.show()
```



6.1 Decision Tree

[189]:
```
from sklearn.tree import DecisionTreeClassifier
D=DecisionTreeClassifier(max_depth=10,criterion="gini",min_samples_split=20)
from sklearn.model_selection import GridSearchCV
params={"max_depth":[10,2,3,1,4,9,8],"criterion":
  ↪["gini","entropy"],"min_samples_split":[11,15,10,8,4,9]}
```

```
G=GridSearchCV(D,param_grid=params,scoring="accuracy",cv=5)
```

[190]:
```
G.fit(X_train,y_train)
```

[190]:
```
GridSearchCV(cv=5,
             estimator=DecisionTreeClassifier(max_depth=10,
                                              min_samples_split=20),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [10, 2, 3, 1, 4, 9, 8],
                         'min_samples_split': [11, 15, 10, 8, 4, 9]},
             scoring='accuracy')
```

[191]:
```
G.best_params_
```

[191]:
```
{'criterion': 'entropy', 'max_depth': 8, 'min_samples_split': 15}
```

[192]:
```
DecisionTree=G.best_estimator_
DecisionTree
```

[192]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=8, min_samples_split=15)
```

[193]:
```
DecisionTree.score(X_train,y_train)*100
```

[193]:
```
97.45058722429103
```

[194]:
```
DecisionTree.score(X_test,y_test)*100
```

[194]:
```
95.1890034364261
```

[195]:
```
pred=DecisionTree.predict(X_test)
pred
```

[195]:
```
array([0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
```

```
0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1,
1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1,
0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1,
0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0,
1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0,
1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,
0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1,
0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1,
0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0,
0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0,
0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,
0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0], dtype=int64)
```

[196]:
```
from sklearn.metrics import␣
↪confusion_matrix,classification_report,accuracy_score
```

[197]:
```
confusion_matrix(y_test,pred)
```

[197]:
```
array([[519,  17],
       [ 25, 312]], dtype=int64)
```

[198]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       536
           1       0.95      0.93      0.94       337

    accuracy                           0.95       873
   macro avg       0.95      0.95      0.95       873
weighted avg       0.95      0.95      0.95       873
```

[199]:
```
accuracy_score(y_test,pred)*100
```

[199]:
```
95.1890034364261
```

[200]:
```
DecisionTree={"DecisionTree":DecisionTreeClassifier()}
```

[201]:
```
DecisionTree.keys()
```

[201]:
```
dict_keys(['DecisionTree'])
```

[202]:
```
DecisionTree.values()
```

[202]:
```
dict_values([DecisionTreeClassifier()])
```

[203]:
```
DecisionTree.items()
```

[203]:
```
dict_items([('DecisionTree', DecisionTreeClassifier())])
```

[204]:
```
Data=[]
for i,j in DecisionTree.items():
    j.fit(X_train,y_train)
    TR=j.score(X_train,y_train)
    TE=j.score(X_test,y_test)
    pred=j.predict(X_test)
    Data.append([i,TR,TE])
    print(i.capitalize())
    print(classification_report(y_test,pred))
    print(confusion_matrix(y_test,pred))
    print("_"*80)
```

```
Decisiontree
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       536
           1       0.94      0.94      0.94       337

    accuracy                           0.95       873
   macro avg       0.95      0.95      0.95       873
weighted avg       0.95      0.95      0.95       873

[[514  22]
 [ 20 317]]
--------------------------------------------------------------------------------
```

[205]:
```
Data
```

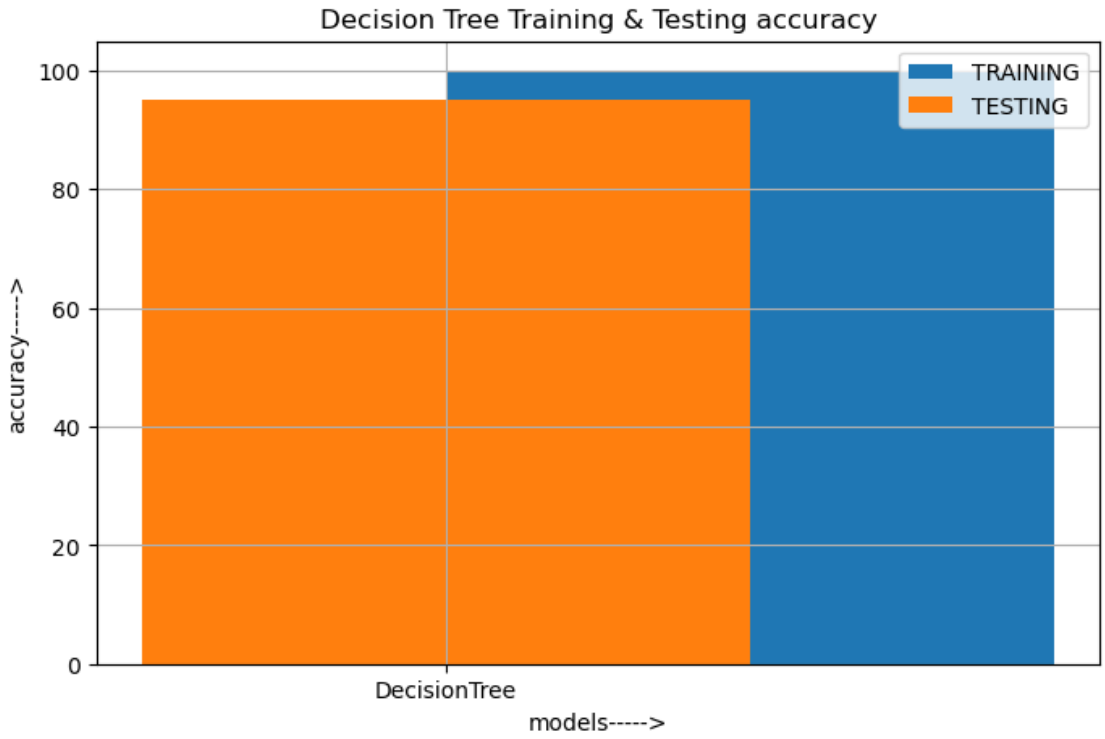
[205]:
```
[['DecisionTree', 1.0, 0.9518900343642611]]
```

[206]:
```
Result=pd.DataFrame(Data,columns=("models name","Training Accuracy","Testing␣
↪Accuracy"))
Result
```

[206]:

	models name	Training Accuracy	Testing Accuracy
0	DecisionTree	1.0	0.95189

[207]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,5))
plt.bar(Result["models name"],Result["Training Accuracy"]*100,width=0.
↪3,align="edge",label="TRAINING")
plt.bar(Result["models name"],Result["Testing Accuracy"]*100,width=0.
↪3,align="center",label="TESTING")
plt.grid()
plt.legend()
plt.xlabel("models----->")
plt.ylabel("accuracy----->")
plt.title("Decision Tree Training & Testing accuracy")
plt.show()
```



6.2 Ensemble Learning

[209]:
```
from sklearn.ensemble import RandomForestClassifier ## RFC are only use␣
 ↪Decision Tree
RFC=RandomForestClassifier(n_estimators=10)
RFC.fit(X_train,y_train)
```

[209]:
```
RandomForestClassifier(n_estimators=10)
```

[210]:
```
RFC.score(X_train,y_train)*100
```

[210]:
```
99.62761386422228
```

[211]:
```
RFC.score(X_test,y_test)*100
```

[211]:
```
95.1890034364261
```

[212]:
```
from sklearn.ensemble import BaggingClassifier
from sklearn.svm import SVC
Bg=BaggingClassifier(estimator=SVC(),n_estimators=10)
```

[213]:
```
Bg.fit(X_train,y_train)
```

[213]:
```
BaggingClassifier(estimator=SVC())
```

[214]:
```
Bg.score(X_train,y_train)*100
```

[214]:
```
61.81609853910054
```

[215]:
```
Bg.score(X_test,y_test)*100
```

[215]:
```
61.39747995418099
```

[216]:
```
from sklearn.ensemble import VotingClassifier
A=VotingClassifier(estimators=[("LogReg",LogisticRegression()),
                               ("SVC",SVC()),
                               ("NB",GaussianNB())])
```

[217]:
```
A.fit(X_train,y_train)
```

[217]:
```
VotingClassifier(estimators=[('LogReg', LogisticRegression()), ('SVC', SVC()),
                             ('NB', GaussianNB())])
```

[218]:
```
A.score(X_train,y_train)*100
```

[218]:
```
86.53680893726727
```

[219]:
```
A.score(X_test,y_test)*100
```

[219]:
```
86.94158075601375
```

[220]:
```
from sklearn.ensemble import StackingClassifier
stacking=StackingClassifier(estimators=[("NB",GaussianNB()),
                                        ("SVC",SVC()),
                                        ("KNN",KNeighborsClassifier()),
                                         ("DT",DecisionTreeClassifier()),
                                         ("LogR",LogisticRegression())],
                            final_estimator=LogisticRegression())
```

[221]:
```
stacking.fit(X_train,y_train)
```

[221]:
```
StackingClassifier(estimators=[('NB', GaussianNB()), ('SVC', SVC()),
                               ('KNN', KNeighborsClassifier()),
                               ('DT', DecisionTreeClassifier()),
                               ('LogR', LogisticRegression())],
                   final_estimator=LogisticRegression())
```

[222]:
```
stacking.score(X_train,y_train)*100
```

[222]:
```
100.0
```

[223]:
```
stacking.score(X_test,y_test)*100
```

[223]:
```
95.53264604810997
```

[224]:
```
models={"RandomForestClassifier":RandomForestClassifier(n_estimators=10),
        "Bagging":BaggingClassifier(estimator=SVC(),n_estimators=10),
        "Voting":VotingClassifier(estimators=[("LogReg",LogisticRegression()),
                                  ("SVC",SVC()),
                                  ("NB",GaussianNB())]),
        "stacking":StackingClassifier(estimators=[("NB",GaussianNB()),
                                  ("SVC",SVC()),
                                  ("KNN",KNeighborsClassifier()),
                                   ("DT",DecisionTreeClassifier()),
                                   ("LogR",LogisticRegression())],
                            final_estimator=LogisticRegression())}
```

[225]:
```
models.values()
```

[225]:
```
dict_values([RandomForestClassifier(n_estimators=10),
BaggingClassifier(estimator=SVC()), VotingClassifier(estimators=[('LogReg',
LogisticRegression()), ('SVC', SVC()),
                                 ('NB', GaussianNB())]),
StackingClassifier(estimators=[('NB', GaussianNB()), ('SVC', SVC()),
```

```
                                    ('KNN', KNeighborsClassifier()),
                                    ('DT', DecisionTreeClassifier()),
                                    ('LogR', LogisticRegression())],
                        final_estimator=LogisticRegression())])
```

[226]:
```
models.keys()
```

[226]:
```
dict_keys(['RandomForestClassifier', 'Bagging', 'Voting', 'stacking'])
```

[227]:
```
models.items()
```

[227]:
```
dict_items([('RandomForestClassifier', RandomForestClassifier(n_estimators=10)),
('Bagging', BaggingClassifier(estimator=SVC())), ('Voting',
VotingClassifier(estimators=[('LogReg', LogisticRegression()), ('SVC', SVC()),
                             ('NB', GaussianNB())])), ('stacking',
StackingClassifier(estimators=[('NB', GaussianNB()), ('SVC', SVC()),
                               ('KNN', KNeighborsClassifier()),
                               ('DT', DecisionTreeClassifier()),
                               ('LogR', LogisticRegression())],
                   final_estimator=LogisticRegression()))])
```

[228]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

[229]:
```
Data=[]
for i,j in models.items():
    j.fit(X_train,y_train)
    TR=j.score(X_train,y_train)
    TE=j.score(X_test,y_test)
    pred=j.predict(X_test)
    Data.append([i,TR,TE])
    print(i.capitalize())
    print(classification_report(y_test,pred))
    print(confusion_matrix(y_test,pred))
    print("_"*80)
```

```
Randomforestclassifier
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       536
           1       0.94      0.90      0.92       337

    accuracy                           0.94       873
   macro avg       0.94      0.93      0.94       873
weighted avg       0.94      0.94      0.94       873

[[516  20]
 [ 33 304]]
```

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

           0       0.61      1.00      0.76       536
           1       0.00      0.00      0.00       337

    accuracy                           0.61       873
   macro avg       0.31      0.50      0.38       873
weighted avg       0.38      0.61      0.47       873

[[536   0]
 [337   0]]
--------------------------------------------------------------------------------
Voting
              precision    recall  f1-score   support

           0       0.90      0.88      0.89       536
           1       0.82      0.85      0.83       337

    accuracy                           0.87       873
   macro avg       0.86      0.87      0.86       873
weighted avg       0.87      0.87      0.87       873

[[474  62]
 [ 52 285]]
--------------------------------------------------------------------------------
Stacking
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       536
           1       0.94      0.94      0.94       337

    accuracy                           0.95       873
   macro avg       0.95      0.95      0.95       873
weighted avg       0.95      0.95      0.95       873

[[515  21]
 [ 20 317]]
--------------------------------------------------------------------------------
```

[230]:
```
Data
```

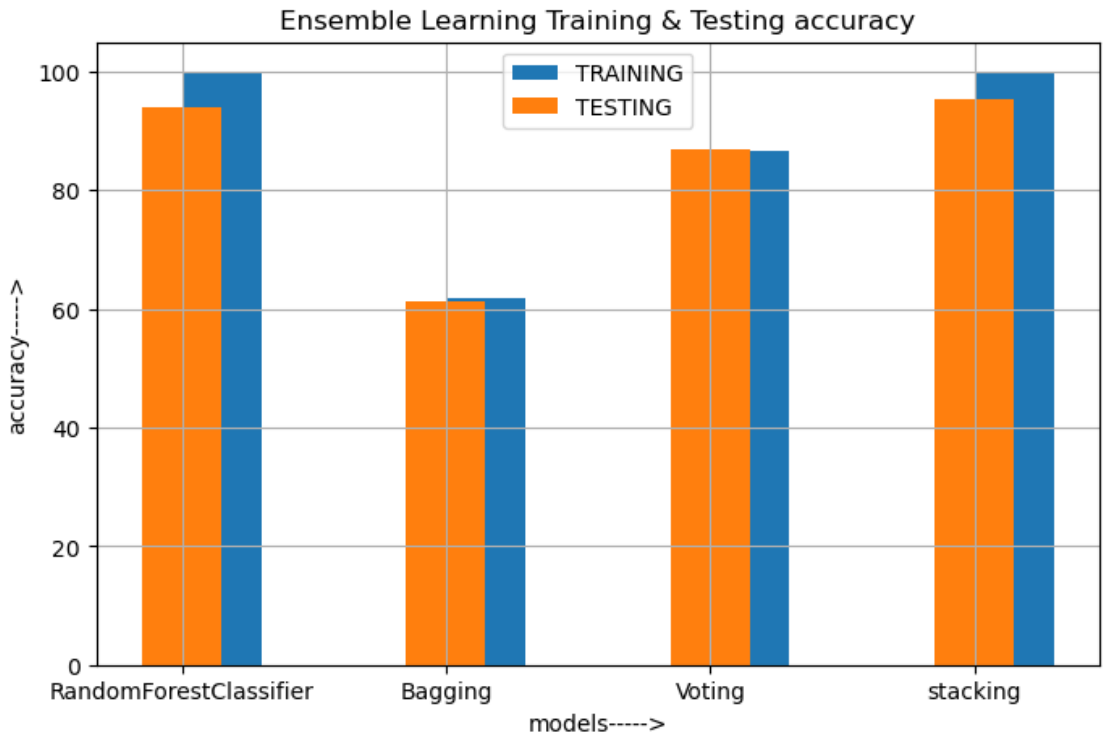
[230]:
```
[['RandomForestClassifier', 0.9977083930105987, 0.9392898052691867],
 ['Bagging', 0.6181609853910054, 0.6139747995418099],
 ['Voting', 0.8653680893726726, 0.8694158075601375],
 ['stacking', 1.0, 0.9530355097365406]]
```

[231]:
```
Result=pd.DataFrame(Data,columns=("models name","Training Accuracy","Testing␣
↪Accuracy"))
Result
```

[231]:

	models name	Training Accuracy	Testing Accuracy
0	RandomForestClassifier	0.997708	0.939290
1	Bagging	0.618161	0.613975
2	Voting	0.865368	0.869416
3	stacking	1.000000	0.953036

[232]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,5))
plt.bar(Result["models name"],Result["Training Accuracy"]*100,width=0.
↪3,align="edge",label="TRAINING")
plt.bar(Result["models name"],Result["Testing Accuracy"]*100,width=0.
↪3,align="center",label="TESTING")
plt.grid()
plt.legend()
plt.xlabel("models----->")
plt.ylabel("accuracy----->")
plt.title("Ensemble Learning Training & Testing accuracy")
plt.show()
```



7 AdaBoost

[234]:
```
from sklearn.ensemble import AdaBoostClassifier
```

[235]:
```
A=AdaBoostClassifier(n_estimators=50)
A.fit(X_train,y_train)
```

[235]:
```
AdaBoostClassifier()
```

[236]:
```
A.score(X_train,y_train)*100
```

[236]:
```
94.35691778859926
```

[237]:
```
A.score(X_test,y_test)*100
```

[237]:
```
94.38717067583046
```

8 PCL

[240]:
```
x=pd.read_csv(r"C:\Users\vidya\Downloads\smart_home_device_usage_data.csv")
x
```

[240]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	\
0	1	Smart Speaker	15.307188	1.961607	
1	2	Camera	19.973343	8.610689	
2	3	Security System	18.911535	2.651777	
3	4	Camera	7.011127	2.341653	
4	5	Camera	22.610684	4.859069	
…	…	…	…	…	
5398	5399	Thermostat	4.556314	5.871764	
5399	5400	Lights	0.561856	1.555992	
5400	5401	Smart Speaker	11.096236	7.677779	
5401	5402	Security System	8.782169	7.467929	
5402	5403	Thermostat	13.540381	9.043076	

	UserPreferences	MalfunctionIncidents	DeviceAgeMonths	\
0	1	4	36	
1	1	0	29	
2	1	0	20	
3	0	3	15	
4	1	3	36	
…	…	…	…	
5398	1	0	28	
5399	1	4	24	
5400	0	0	42	
5401	0	2	28	

```
5402                0                    0               30

      SmartHomeEfficiency
0                       1
1                       1
2                       1
3                       0
4                       1
…                     …
5398                    0
5399                    0
5400                    0
5401                    1
5402                    0

[5403 rows x 8 columns]
```

[242]:
```
x.drop(columns="DeviceType",inplace=True)
x
```

[242]:
```
      UserID  UsageHoursPerDay  EnergyConsumption  UserPreferences  \
0          1         15.307188           1.961607                1
1          2         19.973343           8.610689                1
2          3         18.911535           2.651777                1
3          4          7.011127           2.341653                0
4          5         22.610684           4.859069                1
…       …               …                …               …
5398    5399          4.556314           5.871764                1
5399    5400          0.561856           1.555992                1
5400    5401         11.096236           7.677779                0
5401    5402          8.782169           7.467929                0
5402    5403         13.540381           9.043076                0

      MalfunctionIncidents  DeviceAgeMonths  SmartHomeEfficiency
0                        4               36                    1
1                        0               29                    1
2                        0               20                    1
3                        3               15                    0
4                        3               36                    1
…                      …               …                 …
5398                     0               28                    0
5399                     4               24                    0
5400                     0               42                    0
5401                     2               28                    1
5402                     0               30                    0

[5403 rows x 7 columns]
```

[269]:
```
from sklearn.decomposition import PCA
A=PCA(n_components=5)
x1=A.fit_transform(x)
x2=pd.DataFrame(x1)
A.explained_variance_ratio_.sum()*100
```

[269]: 99.99998135075101

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: