In [1]:
```
import pandas as pd
D=pd.read_csv(r"C:\Users\Admin\Downloads\smart_home_device_usage_data.c
D
```

Out[1]:

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

5403 rows × 8 columns

In [2]:
```
D.columns
```

Out[2]:
```
Index(['UserID', 'DeviceType', 'UsageHoursPerDay', 'EnergyConsumption',
       'UserPreferences', 'MalfunctionIncidents', 'DeviceAgeMonths',
       'SmartHomeEfficiency'],
      dtype='object')
```

In [3]:
```
D.isnull().sum()
```

Out[3]:
```
UserID                  0
DeviceType              0
UsageHoursPerDay        0
EnergyConsumption       0
UserPreferences         0
MalfunctionIncidents    0
DeviceAgeMonths         0
SmartHomeEfficiency     0
dtype: int64
```

In [4]:

```
D.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5403 entries, 0 to 5402
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   UserID                5403 non-null   int64  
 1   DeviceType            5403 non-null   object 
 2   UsageHoursPerDay      5403 non-null   float64
 3   EnergyConsumption     5403 non-null   float64
 4   UserPreferences       5403 non-null   int64  
 5   MalfunctionIncidents  5403 non-null   int64  
 6   DeviceAgeMonths       5403 non-null   int64  
 7   SmartHomeEfficiency   5403 non-null   int64  
dtypes: float64(2), int64(5), object(1)
memory usage: 337.8+ KB
```

In [5]:

```
D['SmartHomeEfficiency'].value_counts()
```

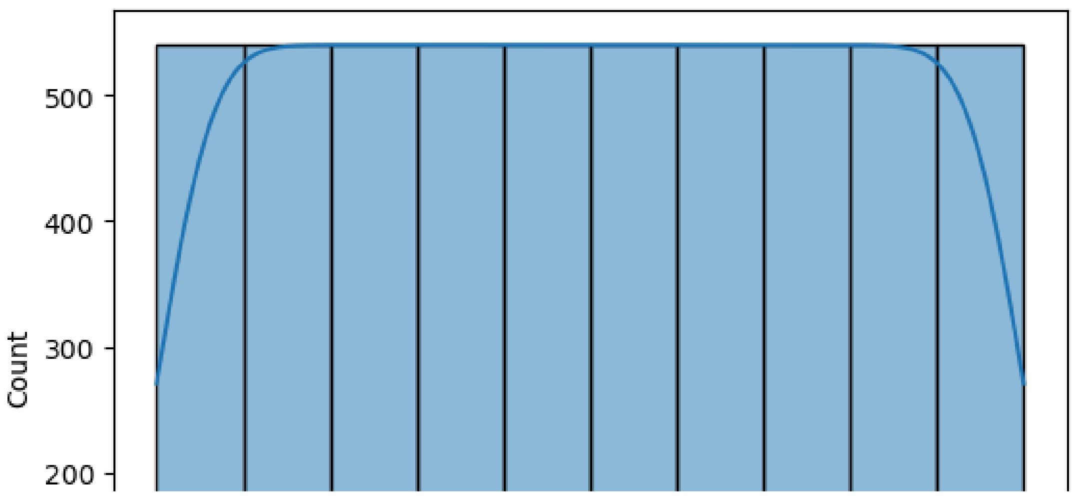
Out[5]:

```
SmartHomeEfficiency
0    3368
1    2035
Name: count, dtype: int64
```

In [6]:

```
import matplotlib.pyplot as plt
import seaborn as sns
for i in D.columns:
    sns.histplot(D[i],bins=10,kde=True)
    plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: 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 [7]:

```
D['UsageHoursPerDay'].mean()
```

Out[7]:

```
12.052992010466317
```

In [8]:
```
D['UsageHoursPerDay'].median()
```
Out[8]:
```
11.903768445051607
```

In [9]:
```
D['UsageHoursPerDay'].mode()[0]
```
Out[9]:
```
0.5012414329089748
```

In [10]:
```
D['EnergyConsumption'].mean()
```
Out[10]:
```
5.054301881355049
```

In [11]:
```
D['EnergyConsumption'].median()
```
Out[11]:
```
5.007047305947374
```

In [12]:
```
D['EnergyConsumption'].mode()[0]
```
Out[12]:
```
0.1015616713227616
```

In [13]:
```
D['DeviceAgeMonths'].mean()
```
Out[13]:
```
30.312233944105127
```

In [14]:
```
D['DeviceAgeMonths'].median()
```
Out[14]:
```
30.0
```

In [15]:
```
D['DeviceAgeMonths'].mode()[0]
```
Out[15]:
```
13
```

In [16]:
```
D['UsageHoursPerDay'].unique()
```
Out[16]:
```
array([15.30718848, 19.97334329, 18.91153466, ..., 11.09623585,
        8.78216919, 13.54038109])
```

In [17]:
```
D['DeviceAgeMonths'].unique()
```
Out[17]:
```
array([36, 29, 20, 15,  3, 56, 53, 23, 58, 54, 46,  9, 30, 19, 38, 50, 48,
       34,  2, 27, 42, 32, 47, 35, 40, 18, 11, 39, 25,  7, 31, 57,  5, 14,
        4, 24, 13, 55, 21, 33, 28, 51, 45, 12, 10, 37,  8,  6, 17,  1, 52,
       26, 22, 59, 41, 16, 44, 49, 43], dtype=int64)
```

In [21]:
```
D['MalfunctionIncidents'].value_counts()
```
Out[21]:
```
MalfunctionIncidents
4    1155
3    1149
0    1049
2    1048
1    1002
Name: count, dtype: int64
```

In [27]:
```
A=D.loc[(D['UsageHoursPerDay']>20)&(D['EnergyConsumption']>7)]
A
```

Out[27]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	UserPreferences	Malfunc
126	127	Camera	22.177206	8.911187	1	
131	132	Camera	23.824094	9.916783	0	
133	134	Camera	21.046222	7.131409	1	
220	221	Thermostat	21.956474	8.249941	0	
246	247	Smart Speaker	21.325025	7.443884	1	
...	...	...	...	...	...	
5242	5243	Lights	22.929521	7.860733	1	
5260	5261	Camera	21.438391	9.998071	1	
5277	5278	Smart Speaker	22.679039	7.713330	0	
5305	5306	Smart Speaker	21.730082	8.502649	1	
5307	5308	Security System	23.593174	7.096208	1	

281 rows × 8 columns

In [28]:
```
A['DeviceAgeMonths'].value_counts().head(5)
```

Out[28]:
```
DeviceAgeMonths
32    9
54    9
56    8
30    8
53    8
Name: count, dtype: int64
```

In [29]:
```
A['MalfunctionIncidents'].value_counts()
```

Out[29]:
```
MalfunctionIncidents
4    72
3    65
0    50
2    48
1    46
Name: count, dtype: int64
```

In [30]:
```
A['SmartHomeEfficiency'].value_counts() #older devices with higher ener
```

Out[30]:
```
SmartHomeEfficiency
0    187
1     94
Name: count, dtype: int64
```

In [32]:
```
S=D.loc[(D['DeviceType']=='Smart Speaker')&(D['EnergyConsumption']>=8)&
S
```

Out[32]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	UserPreferences	Malfunc
50	51	Smart Speaker	15.202794	9.260120	1	
58	59	Smart Speaker	16.163266	9.861676	0	
136	137	Smart Speaker	9.884316	8.494920	0	
194	195	Smart Speaker	12.572135	9.181754	0	
286	287	Smart Speaker	7.836459	8.985337	0	
...	...	...	...	...	...	
5018	5019	Smart Speaker	22.644072	9.314185	1	
5020	5021	Smart Speaker	5.388577	9.696368	0	
5032	5033	Smart Speaker	11.332785	8.078466	0	
5069	5070	Smart Speaker	16.839995	8.603232	0	
5170	5171	Smart Speaker	21.185917	8.206446	0	

100 rows × 8 columns

In [33]:
```
S['SmartHomeEfficiency'].value_counts()
```

Out[33]:
```
SmartHomeEfficiency
0    85
1    15
Name: count, dtype: int64
```

In [34]:
```
S['MalfunctionIncidents'].value_counts()
```

Out[34]:
```
MalfunctionIncidents
0    23
3    21
4    20
1    19
2    17
Name: count, dtype: int64
```

In [39]:

```
E=D.loc[(D['DeviceType']=='Thermostat')&(D['UsageHoursPerDay']>=22)&(D[
E
```

Out[39]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	UserPreferences	Malfunc
138	139	Thermostat	23.488186	5.951844	0	
2211	2212	Thermostat	23.547888	2.386310	0	
3075	3076	Thermostat	23.696852	8.010203	0	
3243	3244	Thermostat	22.723328	2.994062	0	
3779	3780	Thermostat	22.839250	4.432218	0	
4104	4105	Thermostat	23.545233	5.931486	1	
4885	4886	Thermostat	22.445682	8.366797	0	
4925	4926	Thermostat	22.306454	1.926897	1	
4974	4975	Thermostat	22.658449	7.914858	0	
5117	5118	Thermostat	22.803371	6.955159	0	

In [40]:

```
D.groupby('DeviceType')[['UsageHoursPerDay']].mean().reset_index().sort
```

Out[40]:

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

In [199]:
```
R=D.loc[(D['SmartHomeEfficiency']==1)&(D['UserPreferences']==1)]
R
```

Out[199]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	UserPreferences	Malfunc
0	1	Smart Speaker	15.307188	1.961607	1	
1	2	Camera	19.973343	8.610689	1	
2	3	Security System	18.911535	2.651777	1	
4	5	Camera	22.610684	4.859069	1	
5	6	Thermostat	3.422127	5.038625	1	
...	...	...	...	...	...	
5386	5387	Security System	20.393943	3.104494	1	
5387	5388	Lights	10.532275	5.634707	1	
5388	5389	Thermostat	13.472427	6.728036	1	
5393	5394	Security System	18.847219	5.649036	1	
5396	5397	Camera	19.301279	0.792446	1	

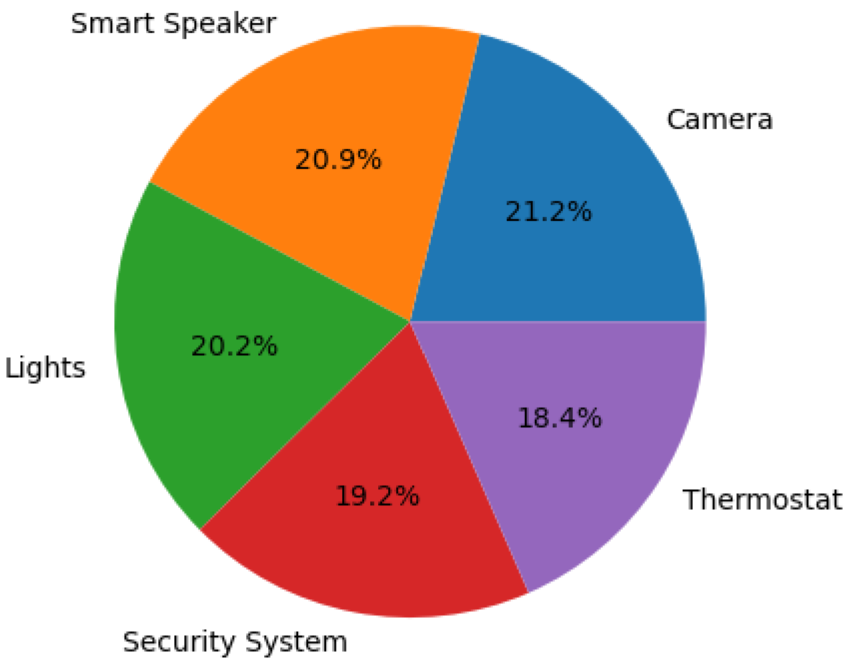
1838 rows × 8 columns

In [200]:
```
R1=R['DeviceType'].value_counts().reset_index()#high user preference is
R1
```

Out[200]:

	DeviceType	count
0	Camera	390
1	Smart Speaker	385
2	Lights	372
3	Security System	352
4	Thermostat	339

In [204]:
```
plt.pie(R1['count'],labels=R1['DeviceType'],autopct='%0.1f%%')
plt.show()
```



In [69]:
```
R['DeviceAgeMonths'].value_counts().head(10)
```

Out[69]:
```
DeviceAgeMonths
13    53
15    53
11    50
23    49
35    49
16    48
24    48
3     47
33    45
25    45
Name: count, dtype: int64
```

In [42]:
```
R['MalfunctionIncidents'].value_counts()
```

Out[42]:
```
MalfunctionIncidents
0    481
4    367
3    365
2    315
1    310
Name: count, dtype: int64
```

In [44]:

```
R.loc[(R['EnergyConsumption']>=8)&(R['DeviceAgeMonths']>=50)]# In old d
```

Out[44]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	UserPreferences	Malfunc
612	613	Camera	8.462352	9.560652	1	
704	705	Lights	17.105197	9.386080	1	
851	852	Camera	11.642712	9.586687	1	
978	979	Security System	17.177277	8.007005	1	
2064	2065	Lights	8.451645	9.869286	1	
2415	2416	Camera	19.948159	9.239124	1	
3280	3281	Thermostat	17.670957	9.096058	1	
3463	3464	Smart Speaker	12.341387	8.432749	1	
4757	4758	Camera	12.352947	9.068957	1	
5090	5091	Camera	11.323301	9.658036	1	
5097	5098	Thermostat	4.427656	8.297744	1	
5376	5377	Lights	19.766901	8.094237	1	

In [70]:

```
R['DeviceType'].value_counts()# from SmartHomeEfficiency & UserPreferen
```

Out[70]:

```
DeviceType
Camera             390
Smart Speaker      385
Lights             372
Security System    352
Thermostat         339
Name: count, dtype: int64
```

In [71]:
```
Z=D.loc[D['SmartHomeEfficiency']==1]
Z
```

Out[71]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	UserPreferences	Malfunc
0	1	Smart Speaker	15.307188	1.961607	1	
1	2	Camera	19.973343	8.610689	1	
2	3	Security System	18.911535	2.651777	1	
4	5	Camera	22.610684	4.859069	1	
5	6	Thermostat	3.422127	5.038625	1	
...	...	...	...	...	...	
5387	5388	Lights	10.532275	5.634707	1	
5388	5389	Thermostat	13.472427	6.728036	1	
5393	5394	Security System	18.847219	5.649036	1	
5396	5397	Camera	19.301279	0.792446	1	
5401	5402	Security System	8.782169	7.467929	0	

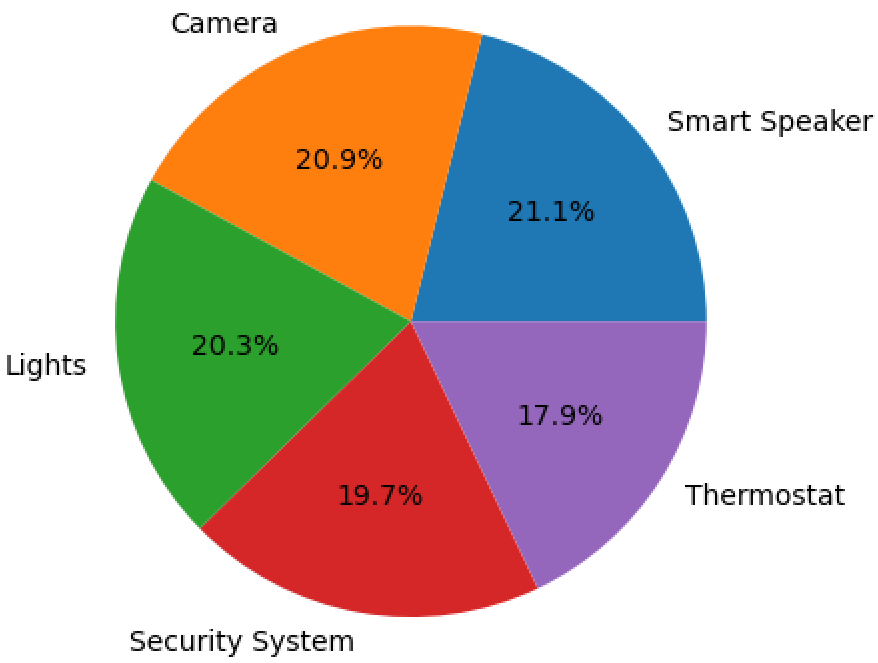
2035 rows × 8 columns

In [203]:
```
z=Z['DeviceType'].value_counts().reset_index()
z
```

Out[203]:

	DeviceType	count
0	Smart Speaker	430
1	Camera	426
2	Lights	413
3	Security System	401
4	Thermostat	365

In [205]:
```
plt.pie(z['count'],labels=z['DeviceType'],autopct='%0.1f%%')
plt.show()
```



In [73]:
```
Z['MalfunctionIncidents'].value_counts()
```

Out[73]:
```
MalfunctionIncidents
0    582
4    394
3    391
2    338
1    330
Name: count, dtype: int64
```

In [74]:
```
Z['DeviceAgeMonths'].value_counts().head(10)
```

Out[74]:
```
DeviceAgeMonths
13    58
15    58
11    56
35    54
3     53
24    53
25    51
26    51
16    51
23    50
Name: count, dtype: int64
```

In [75]:
```
Z1=Z.loc[Z['DeviceType']=='Smart Speaker']
Z1
```

Out[75]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	UserPreferences	Malfunc
0	1	Smart Speaker	15.307188	1.961607	1	
14	15	Smart Speaker	22.494525	1.468928	0	
16	17	Smart Speaker	11.810032	8.228216	1	
19	20	Smart Speaker	1.018554	1.344045	1	
22	23	Smart Speaker	19.638089	4.749577	1	
...	...	...	...	...	...	
5366	5367	Smart Speaker	4.257961	0.527909	1	
5368	5369	Smart Speaker	14.129244	0.860810	0	
5370	5371	Smart Speaker	2.115647	6.732271	1	
5381	5382	Smart Speaker	9.502884	6.430738	1	
5383	5384	Smart Speaker	23.229510	4.061440	0	

430 rows × 8 columns

In [76]:
```
Z1['DeviceAgeMonths'].value_counts().head(5)
```

Out[76]:
```
DeviceAgeMonths
25    17
13    14
35    13
11    12
18    12
Name: count, dtype: int64
```

In [77]:
```
Z1['MalfunctionIncidents'].value_counts()
```

Out[77]:
```
MalfunctionIncidents
0    121
4     91
3     83
2     76
1     59
Name: count, dtype: int64
```

From above we can say that Smart Speaker is most efficient one with less age and malfunction incidents.

In [207]:
```
B=D.loc[D['SmartHomeEfficiency']==0]
B
```

Out[207]:

	UserID	DeviceType	UsageHoursPerDay	EnergyConsumption	UserPreferences	Malfunc
3	4	Camera	7.011127	2.341653	0	
6	7	Security System	21.065640	2.229344	0	
7	8	Security System	23.317096	2.791421	0	
9	10	Camera	17.468553	7.212756	1	
10	11	Smart Speaker	1.446710	7.723881	0	
...	...	...	...	...	...	
5397	5398	Lights	8.633520	4.249140	0	
5398	5399	Thermostat	4.556314	5.871764	1	
5399	5400	Lights	0.561856	1.555992	1	
5400	5401	Smart Speaker	11.096236	7.677779	0	
5402	5403	Thermostat	13.540381	9.043076	0	

3368 rows × 8 columns

In [210]:
```
B['DeviceAgeMonths'].value_counts().head(5)#old devices with high malfu
```

Out[210]:
```
DeviceAgeMonths
53    97
42    84
59    82
44    79
38    79
Name: count, dtype: int64
```

In [212]:
```
B1=B['DeviceType'].value_counts().reset_index()
B1
```

Out[212]:

	DeviceType	count
0	Smart Speaker	678
1	Camera	675
2	Thermostat	674
3	Lights	674
4	Security System	667

In [215]:
```
B['MalfunctionIncidents'].value_counts()
```

Out[215]:
```
MalfunctionIncidents
4    761
3    758
2    710
1    672
0    467
Name: count, dtype: int64
```

In [79]:
```
DS=pd.get_dummies(D['DeviceType'],drop_first=True).replace({True:1,Fals
DS
```

Out[79]:

	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 × 4 columns

In [80]:
```
N=pd.concat([D,DS],axis=1)
N
```

Out[80]:

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

5403 rows × 12 columns

In [81]:
```
N.drop(columns="DeviceType",inplace=True)
```

In [82]:
```
N.drop(columns="UserID",inplace=True)
```

In [83]:
```
N
```

Out[83]:

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

5403 rows × 10 columns

In [84]:
```
FS=N.drop(columns='SmartHomeEfficiency',axis=1)
T=N['SmartHomeEfficiency']
```

In [85]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(FS,T,train_size=0.65,rar
```

In [92]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
Log=LogisticRegression()
params={"C":[0.2,0.4,0.02,0.8],"penalty":["l1","l2"]}
G=GridSearchCV(Log,param_grid=params,scoring="accuracy",cv=6)
```

In [93]:
```
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 [94]:
```
G.best_params_
```

Out[94]: `{'C': 0.2, 'penalty': 'l2'}`

In [95]:
```
model=G.best_estimator_
model
```

Out[95]: `LogisticRegression(C=0.2)`

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]:
```
pred=model.predict(X_test)
pred
```

Out[96]: `array([0, 0, 0, ..., 0, 0, 0], dtype=int64)`

In [97]:
```
model.score(X_train,y_train)
```

Out[97]: `0.8786670464255198`

In [98]:
```
model.score(X_test,y_test)
```

Out[98]: `0.86892177589852`

In [99]:
```
from sklearn.metrics import classification_report,accuracy_score,confus
```

In [100]:
```
accuracy_score(y_test,pred)
```

Out[100]: `0.86892177589852`

In [101]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.89      0.90      1195
           1       0.82      0.83      0.82       697

    accuracy                           0.87      1892
   macro avg       0.86      0.86      0.86      1892
weighted avg       0.87      0.87      0.87      1892
```

In [102]:
```
confusion_matrix(y_test,pred)
```

Out[102]:
```
array([[1065,  130],
       [ 118,  579]], dtype=int64)
```

In [125]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
SVC=SVC()
prm={"gamma":[0.4,0.6,0.2,0.8],"kernel":['linear','rbf']}
g=GridSearchCV(SVC,param_grid=prm,scoring='accuracy',cv=6)
```

In [126]:
```
g.fit(X_train,y_train)
```

Out[126]:
```
GridSearchCV(cv=6, estimator=SVC(),
             param_grid={'gamma': [0.4, 0.6, 0.2, 0.8],
                         'kernel': ['linear', 'rbf']},
             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 [127]:
```
g.best_params_
```

Out[127]:
```
{'gamma': 0.4, 'kernel': 'linear'}
```

In [128]:
```
L=g.best_estimator_
L
```

Out[128]:
```
SVC(gamma=0.4, kernel='linear')
```

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]:
```
predict=L.predict(X_test)
predict
```

Out[129]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [130]:
```
L.score(X_train,y_train)
```
Out[130]: 0.877527769866135

In [131]:
```
L.score(X_test,y_test)
```
Out[131]: 0.86892177589852

In [132]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
KNC=KNeighborsClassifier()
params={'n_neighbors':[4,6,5,10]}
I=GridSearchCV(KNC,param_grid=params,scoring="accuracy",cv=7)
```

In [133]:
```
I.fit(X_train,y_train)
```
Out[133]:
```
GridSearchCV(cv=7, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': [4, 6, 5, 10]}, 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 [134]:
```
I.best_params_
```
Out[134]: {'n_neighbors': 6}

In [135]:
```
J=I.best_estimator_
J
```
Out[135]: KNeighborsClassifier(n_neighbors=6)

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]:
```
P=J.predict(X_test)
P
```
Out[136]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)

In [137]:
```
J.score(X_train,y_train)
```
Out[137]: 0.8057533466248932

In [138]:
```
J.score(X_test,y_test)
```
Out[138]: 0.7066596194503171

In [143]:
```
from sklearn.ensemble import RandomForestClassifier
R1= RandomForestClassifier(n_estimators=25)
```

In [144]:
```
R1.fit(X_train,y_train)
```

Out[144]:
```
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 [145]:
```
R1.score(X_train,y_train)
```

Out[145]:
```
0.9980062660210767
```

In [146]:
```
R1.score(X_test,y_test)
```

Out[146]:
```
0.9513742071881607
```

In [147]:
```
from sklearn.ensemble import AdaBoostClassifier
AD=AdaBoostClassifier(n_estimators=50)
AD.fit(X_train,y_train)
```

Out[147]:
```
AdaBoostClassifier()
```

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 [148]:
```
AD.score(X_train,y_train)
```

Out[148]:
```
0.9376246083736827
```

In [149]:
```
AD.score(X_test,y_test)
```

Out[149]:
```
0.9312896405919662
```

In [158]:
```
H={'models':["Log","SVC","KNN","R1","AD"],"Train":[87.86,87.75,80.57,9
H
```

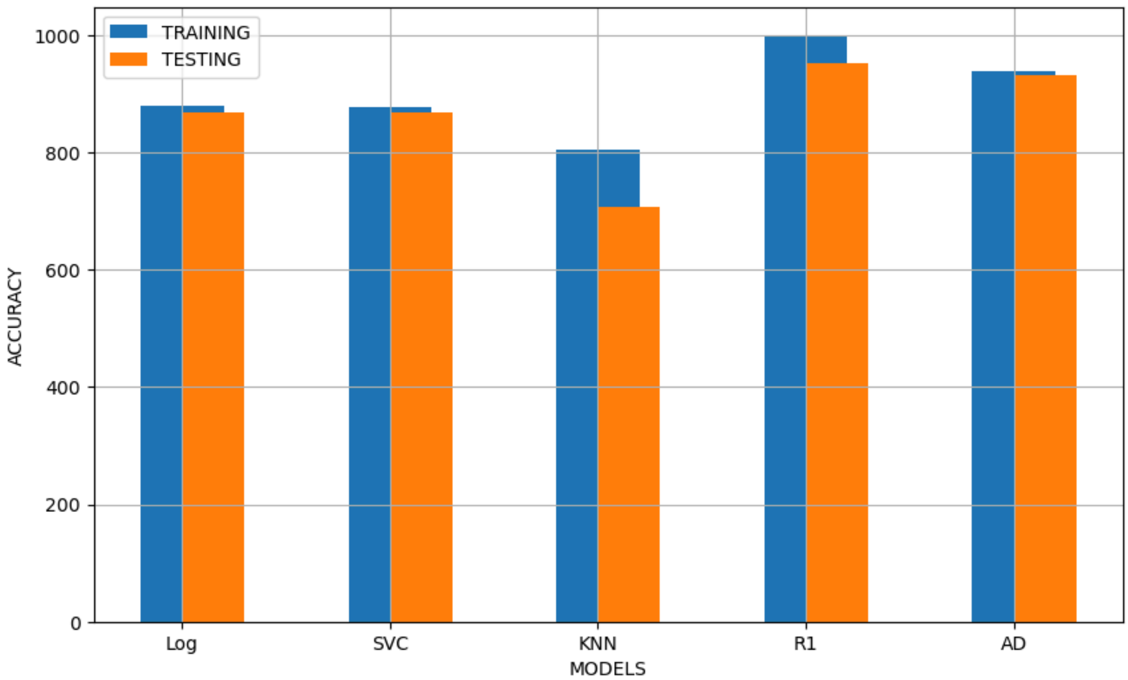
Out[158]:
```
{'models': ['Log', 'SVC', 'KNN', 'R1', 'AD'],
 'Train': [87.86, 87.75, 80.57, 99.8, 93.76],
 'Test': [86.89, 86.89, 70.66, 95.13, 93.12]}
```

In [159]:
```
H=pd.DataFrame(H)
H
```

Out[159]:

	models	Train	Test
0	Log	87.86	86.89
1	SVC	87.75	86.89
2	KNN	80.57	70.66
3	R1	99.80	95.13
4	AD	93.76	93.12

In [160]:
```
plt.figure(figsize=(10,6))
plt.bar(H['models'],H['Train']*10,align='center',width=0.4,label='TRAIN
plt.bar(H['models'],H['Test']*10,align='edge',width=0.3,label='TESTING'
plt.grid()
plt.legend()
plt.xlabel('MODELS')
plt.ylabel('ACCURACY')
plt.show()
```



In [75]:
```
#from above plot,Logistic Regression , RandomForest and AdaBoost works
```

In [161]:
```
from sklearn.naive_bayes import GaussianNB   , BernoulliNB   ,ComplementN

Gn=GaussianNB()
Gn.fit(X_train,y_train)
```

Out[161]:
```
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 [162]:
```
models={'Gaussian':GaussianNB(),
       'Bernoulian':BernoulliNB(),
       'Complement':ComplementNB(),
       'Categorical':CategoricalNB()}
```

In [163]:
```
models.keys()
```

Out[163]:
```
dict_keys(['Gaussian', 'Bernoulian', 'Complement', 'Categorical'])
```

In [164]:
```
models.values()
```

Out[164]:
```
dict_values([GaussianNB(), BernoulliNB(), ComplementNB(), CategoricalNB
()])
```

In [165]:
```
from sklearn.metrics import classification_report,accuracy_score,confus
```

In [166]:

```
Data1=[]

for k,l in models.items():
    l.fit(X_train,y_train)
    TR=l.score(X_train,y_train)
    TE=l.score(X_test,y_test)
    pred=l.predict(X_test)


    Data1.append([k,TR,TE])
    print(k.upper())
    print(classification_report(y_test,pred))
    print(confusion_matrix(y_test,pred))
    print('__'*40)
```

```
GAUSSIAN
              precision    recall  f1-score   support

           0       0.93      0.77      0.84      1195
           1       0.70      0.90      0.79       697

    accuracy                           0.82      1892
   macro avg       0.81      0.84      0.81      1892
weighted avg       0.84      0.82      0.82      1892

[[920 275]
 [ 69 628]]
```

```
BERNOULIAN
              precision    recall  f1-score   support

           0       0.93      0.72      0.81      1195
           1       0.65      0.90      0.76       697

    accuracy                           0.79      1892
   macro avg       0.79      0.81      0.78      1892
weighted avg       0.83      0.79      0.79      1892

[[859 336]
 [ 68 629]]
```

```
COMPLEMENT
              precision    recall  f1-score   support

           0       0.80      0.68      0.73      1195
           1       0.56      0.70      0.62       697

    accuracy                           0.69      1892
   macro avg       0.68      0.69      0.68      1892
weighted avg       0.71      0.69      0.69      1892

[[815 380]
 [210 487]]
```

```
CATEGORICAL
              precision    recall  f1-score   support

           0       0.92      0.94      0.93      1195
           1       0.89      0.86      0.87       697

    accuracy                           0.91      1892
   macro avg       0.90      0.90      0.90      1892
weighted avg       0.91      0.91      0.91      1892

[[1119   76]
 [  96  601]]
```

In [167]:
```
Data1
```

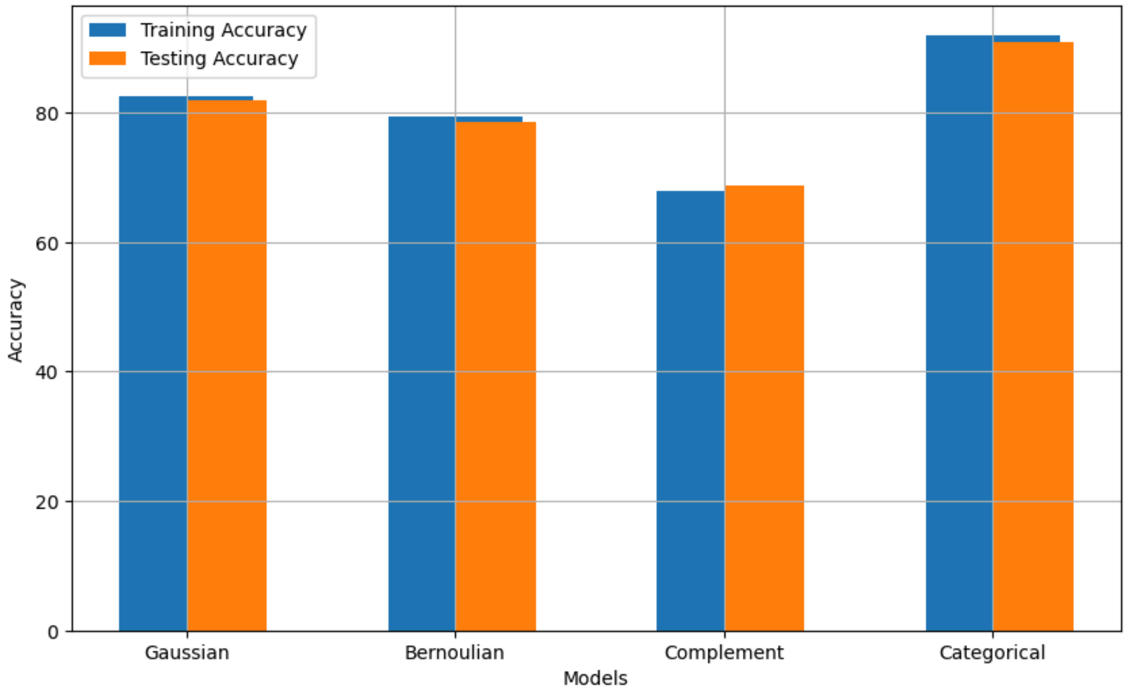
Out[167]:
```
[['Gaussian', 0.825690686414127, 0.8181818181818182],
 ['Bernoulian', 0.7949302193107377, 0.7864693446088795],
 ['Complement', 0.6792936485331814, 0.6881606765327696],
 ['Categorical', 0.9199658217032185, 0.9090909090909091]]
```

In [168]:
```
R=pd.DataFrame(Data1,columns=('model name','Train','Test'))
R
```

Out[168]:

	model name	Train	Test
0	Gaussian	0.825691	0.818182
1	Bernoulian	0.794930	0.786469
2	Complement	0.679294	0.688161
3	Categorical	0.919966	0.909091

In [169]:
```
plt.figure(figsize=(10,6))
plt.bar(R['model name'],R['Train']*100,align='center',width=0.5,label=
plt.bar(R['model name'],R['Test']*100,align='edge',width=0.3,label='Tes
plt.grid()
plt.legend()
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.show()
```



#ANN

In [170]:
```
!pip install tensorflow
```

```
Defaulting to user installation because normal site-packages is not writea
ble
Requirement already satisfied: tensorflow in c:\users\admin\appdata\roamin
g\python\python311\site-packages (2.17.0)
Requirement already satisfied: tensorflow-intel==2.17.0 in c:\users\admin
\appdata\roaming\python\python311\site-packages (from tensorflow) (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\admin\appdata\ro
aming\python\python311\site-packages (from tensorflow-intel==2.17.0->tenso
rflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\admin\appdata
\roaming\python\python311\site-packages (from tensorflow-intel==2.17.0->te
nsorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in c:\users\admin\appd
ata\roaming\python\python311\site-packages (from tensorflow-intel==2.17.0-
>tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\u
sers\admin\appdata\roaming\python\python311\site-packages (from tensorflow
-intel==2.17.0->tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\admin\appda
ta\roaming\python\python311\site-packages (from tensorflow-intel==2.17.0->
tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in c:\users\admin\appdata\roam
ing\python\python311\site-packages (from tensorflow-intel==2.17.0->tensorf
low) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\admin\appdata
\roaming\python\python311\site-packages (from tensorflow-intel==2.17.0->te
nsorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in c:\users\admin\a
ppdata\roaming\python\python311\site-packages (from tensorflow-intel==2.1
7.0->tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\admin\appdata
\roaming\python\python311\site-packages (from tensorflow-intel==2.17.0->te
nsorflow) (3.4.0)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.17.0->tensorflow) (23.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.
3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\programdata\anaconda3\lib\sit
e-packages (from tensorflow-intel==2.17.0->tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\programdata\anaco
nda3\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow) (2.31.
0)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib
\site-packages (from tensorflow-intel==2.17.0->tensorflow) (68.2.2)
Requirement already satisfied: six>=1.12.0 in c:\programdata\anaconda3\lib
\site-packages (from tensorflow-intel==2.17.0->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\admin\appdata
\roaming\python\python311\site-packages (from tensorflow-intel==2.17.0->te
nsorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\programdata
\anaconda3\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow)
(4.9.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\programdata\anaconda3\l
ib\site-packages (from tensorflow-intel==2.17.0->tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\admin\appda
ta\roaming\python\python311\site-packages (from tensorflow-intel==2.17.0->
tensorflow) (1.66.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in c:\users\admin\a
ppdata\roaming\python\python311\site-packages (from tensorflow-intel==2.1
7.0->tensorflow) (2.17.1)
Requirement already satisfied: keras>=3.2.0 in c:\users\admin\appdata\roam
ing\python\python311\site-packages (from tensorflow-intel==2.17.0->tensorf
```

```
low) (3.5.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\admin\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.17.0->tensorflow) (0.31.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\programdata\anaconda3\lib\site-packages (from tensorflow-intel==2.17.0->tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\programdata\anaconda3\lib\site-packages (from astunparse>=1.6.0->tensorflow-intel==2.17.0->tensorflow) (0.41.2)
Requirement already satisfied: rich in c:\programdata\anaconda3\lib\site-packages (from keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (13.3.5)
Requirement already satisfied: namex in c:\users\admin\appdata\roaming\python\python311\site-packages (from keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in c:\users\admin\appdata\roaming\python\python311\site-packages (from keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (0.12.1)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\programdata\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2024.2.2)
Requirement already satisfied: markdown>=2.6.8 in c:\programdata\anaconda3\lib\site-packages (from tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\admin\appdata\roaming\python\python311\site-packages (from tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (2.2.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\programdata\anaconda3\lib\site-packages (from werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (2.1.3)
Requirement already satisfied: markdown-it-py<3.0.0,>=2.2.0 in c:\programdata\anaconda3\lib\site-packages (from rich->keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\programdata\anaconda3\lib\site-packages (from rich->keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\programdata\anaconda3\lib\site-packages (from markdown-it-py<3.0.0,>=2.2.0->rich->keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (0.1.0)
```

In [171]:

```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout
```

In [172]:
```
model= Sequential([

    Dense(60,input_shape=(X_train.shape[1],),activation='relu'),#input
    Dense(30,activation='relu'),#hidden layer
    Dropout(0.2),
    Dense(30,activation='relu'),
    Dense(30,activation='relu'),
    Dropout(0.2),
    Dense(30,activation='relu'),#hidden layer
    Dense(1,activation='sigmoid'),#output layer
])
```

```
C:\Users\Admin\AppData\Roaming\Python\Python311\site-packages\keras\src\la
yers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Inp
ut(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [173]:
```
model.compile(
    loss='binary_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```

In [175]:
```
model.fit(X_train,y_train,epochs=10,validation_split=0.2)
```

```
Epoch 1/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.8733 - loss: 0.3410 -
val_accuracy: 0.8933 - val_loss: 0.2845
Epoch 2/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8784 - loss: 0.3386 -
val_accuracy: 0.8549 - val_loss: 0.3435
Epoch 3/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8619 - loss: 0.3586 -
val_accuracy: 0.9018 - val_loss: 0.2855
Epoch 4/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8821 - loss: 0.3372 -
val_accuracy: 0.8933 - val_loss: 0.2870
Epoch 5/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8826 - loss: 0.3230 -
val_accuracy: 0.8762 - val_loss: 0.3137
Epoch 6/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8836 - loss: 0.3187 -
val_accuracy: 0.8962 - val_loss: 0.2921
Epoch 7/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8719 - loss: 0.3336 -
val_accuracy: 0.8962 - val_loss: 0.2848
Epoch 8/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8893 - loss: 0.3173 -
val_accuracy: 0.8947 - val_loss: 0.2873
Epoch 9/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8863 - loss: 0.3141 -
val_accuracy: 0.8890 - val_loss: 0.3006
Epoch 10/10
88/88 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8844 - loss: 0.3181 -
val_accuracy: 0.8933 - val_loss: 0.2934
```

Out[175]:
```
<keras.src.callbacks.history.History at 0x277bf749250>
```

In [176]:
```
pred=model.predict(X_test)
pred
```

```
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```

Out[176]:
```
array([[0.08804366],
       [0.18649071],
       [0.67115134],
       ...,
       [0.08145802],
       [0.1564011 ],
       [0.11971354]], dtype=float32)
```

In [177]:
```
pred1=[]
for i in pred:
    if i<=0.5:
        pred1.append(0)
    else:
        pred1.append(1)
```

In [178]:
```
pred1
```

Out[178]:
```
[0,
 0,
 1,
 0,
 0,
 0,
 0,
 0,
 1,
 0,
 0,
 0,
 0,
 0,
 0,
 1,
 0,
 0,
 0,
```

In [179]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classificat
confusion_matrix(y_test,pred1)
```

Out[179]:
```
array([[1123,   72],
       [ 140,  557]], dtype=int64)
```

In [180]:
```
accuracy_score(y_test,pred1)
```

Out[180]:
```
0.8879492600422833
```

In [181]:
```
print(classification_report(y_test,pred1))
```

```
              precision    recall  f1-score   support

           0       0.89      0.94      0.91      1195
           1       0.89      0.80      0.84       697

    accuracy                           0.89      1892
   macro avg       0.89      0.87      0.88      1892
weighted avg       0.89      0.89      0.89      1892
```

```
                    #Project Report

    #The goal is to predict the efficiency of smart home devices based
    on the available features like EnergyConsumption,UsageHoursPerDay,
    and DeviceAgeMonths.
    #This helps identify underperforming devices and optimize their us
    e,leading to better energy consumption and user satisfaction.
```

#Discussion: #•Insights: The analysis shows that older devices with higher energy consumption and more malfunction incidents tend to be less efficient. #•Model Comparison: Both ML models and ANN performed well, but the ANN showed better generalization on unseen data. #The dataset could benefit from additional features such as device maintenance history,Brands or more granular usage data.

#Conclusion : #This project successfully developed predictive models to estimate smart home device efficiency. #Using machine learning and ANN, we demonstrated how energy consumption, usage, and other device features impact efficiency. #The findings can help users optimize their smart home devices for better performance and energy savings.

In [ ]:
```

```