In [30]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import cross_val_score
import warnings
warnings.filterwarnings('ignore')
```

In [76]:

```
salary=pd.read_csv('salary.csv')
```

In [77]:

```
salary
```

Out[77]:

	salary	experience	education	management
0	13876	1	Bachelor	Y
1	11608	1	Ph.D	N
2	18701	1	Ph.D	Y
3	11283	1	Master	N
4	11767	1	Ph.D	N
5	20872	2	Master	Y
6	11772	2	Master	N
7	10535	2	Bachelor	N
8	12195	2	Ph.D	N
9	12313	3	Master	N
10	14975	3	Bachelor	Y
11	21371	3	Master	Y
12	19800	3	Ph.D	Y
13	11417	4	Bachelor	N
14	20263	4	Ph.D	Y
15	13231	4	Ph.D	N
16	12884	4	Master	N
17	13245	5	Master	N
18	13677	5	Ph.D	N
19	15965	5	Bachelor	Y
20	12336	6	Bachelor	N
21	21352	6	Ph.D	Y
22	13839	6	Master	N
23	22884	6	Master	Y
24	16978	7	Bachelor	Y
25	14803	8	Master	N
26	17404	8	Bachelor	Y
27	22184	8	Ph.D	Y

	salary	experience	education	management
28	13348	8	Bachelor	N
29	14467	10	Bachelor	N
30	15942	10	Master	N
31	23174	10	Ph.D	Y
32	23780	10	Master	Y
33	25410	11	Master	Y
34	14861	11	Bachelor	N
35	16882	12	Master	N
36	24170	12	Ph.D	Y
37	15990	13	Bachelor	N
38	26330	13	Master	Y
39	17949	14	Master	N
40	25685	15	Ph.D	Y
41	27837	16	Master	Y
42	18838	16	Master	N
43	17483	16	Bachelor	N
44	19207	17	Master	N
45	19346	20	Bachelor	N

In [78]:

```
salary.describe()
```

Out[78]:

	salary	experience
count	46.000000	46.000000
mean	17270.195652	7.500000
std	4716.631513	5.171503
min	10535.000000	1.000000
25%	13320.750000	3.000000
50%	16436.000000	6.000000
75%	20719.750000	11.000000
max	27837.000000	20.000000

In [79]:

```
salary.dtypes
```

Out[79]:

```
salary          int64
experience      int64
education      object
management     object
dtype: object
```

In [81]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
list1=['salary','experience','education','management']
for val in list1:
    salary[val]=le.fit_transform(salary[val].astype(str))
```

In [82]:

```
salary
```

```
salary
```

Out[82]:

	salary	experience	education	management
0	15	0	0	1
1	3	0	2	0
2	28	0	2	1
3	1	0	1	0
4	4	0	2	0
5	34	9	1	1
6	5	9	1	0
7	0	9	0	0
8	6	9	2	0
9	7	11	1	0
10	19	11	0	1
11	36	11	1	1
12	32	11	2	1
13	2	12	0	0
14	33	12	2	1
15	10	12	2	0
16	9	12	1	0
17	11	13	1	0
18	13	13	2	0
19	21	13	0	1
20	8	14	0	0
21	35	14	2	1
22	14	14	1	0
23	38	14	1	1
24	24	15	0	1
25	17	16	1	0
26	25	16	0	1
27	37	16	2	1
28	12	16	0	0
29	16	1	0	0
30	20	1	1	0
31	39	1	2	1
32	40	1	1	1
33	42	2	1	1
34	18	2	0	0
35	23	3	1	0
36	41	3	2	1
37	22	4	0	0
38	44	4	1	1
39	27	5	1	0
40	43	6	2	1
41	45	7	1	1
42	29	7	1	0
43	26	7	0	0
44	30	8	1	0
45	31	10	0	0

In [83]:

```
sns.heatmap(salary.isnull())
```

Out[83]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21a47e776d0>
```

In [84]:

```
salary.isnull().sum()
```

Out[84]:

```
salary         0
experience     0
education      0
management     0
dtype: int64
```

In [85]:

```
salary.corr()
```

Out[85]:

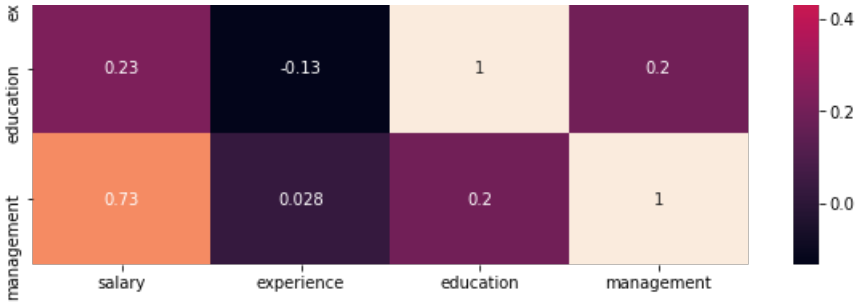
	salary	experience	education	management
salary	1.000000	-0.080479	0.229857	0.730005
experience	-0.080479	1.000000	-0.131927	0.027792
education	0.229857	-0.131927	1.000000	0.196684
management	0.730005	0.027792	0.196684	1.000000

In [86]:

```
plt.figure(figsize=(10,6))
sns.heatmap(salary.corr(),annot=True)
```

Out[86]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21a478d27c0>
```



In [87]:

```
salary.skew()
```

Out[87]:

```
salary         0.000000
experience    -0.189173
education      0.038033
management     0.272071
dtype: float64
```

In [88]:

```
from scipy.stats import zscore
z_score=abs(zscore(salary))
print(salary.shape)
sal=salary.loc[(z_score<3).all(axis=1)]
print(sal.shape)
```

```
(46, 4)
(46, 4)
```

In [89]:

```
sal
```

Out[89]:

	salary	experience	education	management
0	15	0	0	1
1	3	0	2	0
2	28	0	2	1
3	1	0	1	0
4	4	0	2	0
5	34	9	1	1
6	5	9	1	0
7	0	9	0	0
8	6	9	2	0
9	7	11	1	0
10	19	11	0	1
11	36	11	1	1
12	32	11	2	1
13	2	12	0	0
14	33	12	2	1
15	10	12	2	0
16	9	12	1	0
17	11	13	1	0
18	13	13	2	0

	salary	experience	education	management
19	21	13	0	1
20	8	14	0	0
21	35	14	2	1
22	14	14	1	0
23	38	14	1	1
24	24	15	0	1
25	17	16	1	0
26	25	16	0	1
27	37	16	2	1
28	12	16	0	0
29	16	1	0	0
30	20	1	1	0
31	39	1	2	1
32	40	1	1	1
33	42	2	1	1
34	18	2	0	0
35	23	3	1	0
36	41	3	2	1
37	22	4	0	0
38	44	4	1	1
39	27	5	1	0
40	43	6	2	1
41	45	7	1	1
42	29	7	1	0
43	26	7	0	0
44	30	8	1	0
45	31	10	0	0

In [90]:

```
x=sal.iloc[:,0:-1]
```

In [91]:

```
x
```

Out[91]:

	salary	experience	education
0	15	0	0
1	3	0	2
2	28	0	2
3	1	0	1
4	4	0	2
5	34	9	1
6	5	9	1
7	0	9	0
8	6	9	2
9	7	11	1
10	19	11	0
11	36	11	1
12	32	11	2
13	3	12	0

	salary	experience	education
14	33	12	2
15	10	12	2
16	9	12	1
17	11	13	1
18	13	13	2
19	21	13	0
20	8	14	0
21	35	14	2
22	14	14	1
23	38	14	1
24	24	15	0
25	17	16	1
26	25	16	0
27	37	16	2
28	12	16	0
29	16	1	0
30	20	1	1
31	39	1	2
32	40	1	1
33	42	2	1
34	18	2	0
35	23	3	1
36	41	3	2
37	22	4	0
38	44	4	1
39	27	5	1
40	43	6	2
41	45	7	1
42	29	7	1
43	26	7	0
44	30	8	1
45	31	10	0

In [92]:

```
x.shape
```

Out[92]:

```
(46, 3)
```

In [93]:

```
y=sal.iloc[:,-1]
```

In [94]:

```
y
```

Out[94]:

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

```
5     1
6     0
7     0
8     0
9     0
10    1
11    1
12    1
13    0
14    1
15    0
16    0
17    0
18    0
19    1
20    0
21    1
22    0
23    1
24    1
25    0
26    1
27    1
28    0
29    0
30    0
31    1
32    1
33    1
34    0
35    0
36    1
37    0
38    1
39    0
40    1
41    1
42    0
43    0
44    0
45    0
Name: management, dtype: int32
```

In [95]:

```
y.shape
```

Out[95]:

```
(46,)
```

In [96]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.22,random_state=50)
```

In [97]:

```
lr=LogisticRegression()
```

In [98]:

```
lr.fit(x_train,y_train)
lr.score(x_train,y_train)
pred=lr.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.7272727272727273
[[5 3]
 [0 3]]
              precision    recall  f1-score   support
```

```
              -                                   --

           0       1.00      0.62      0.77         8
           1       0.50      1.00      0.67         3

    accuracy                           0.73        11
   macro avg       0.75      0.81      0.72        11
weighted avg       0.86      0.73      0.74        11
```

In [99]:

```
gnb=GaussianNB()
gnb.fit(x_train,y_train)
gnb.score(x_train,y_train)
predgnb=gnb.predict(x_test)
print(accuracy_score(y_test,predgnb))
print(confusion_matrix(y_test,predgnb))
print(classification_report(y_test,predgnb))
```

```
0.6363636363636364
[[5 3]
 [1 2]]
              precision    recall  f1-score   support

           0       0.83      0.62      0.71         8
           1       0.40      0.67      0.50         3

    accuracy                           0.64        11
   macro avg       0.62      0.65      0.61        11
weighted avg       0.72      0.64      0.66        11
```

In [100]:

```
svc=SVC(kernel='rbf')
svc.fit(x_train,y_train)
svc.score(x_train,y_train)
predsvc=svc.predict(x_test)
print(accuracy_score(y_test,predsvc))
print(confusion_matrix(y_test,predsvc))
print(classification_report(y_test,predsvc))
```

```
0.8181818181818182
[[6 2]
 [0 3]]
              precision    recall  f1-score   support

           0       1.00      0.75      0.86         8
           1       0.60      1.00      0.75         3

    accuracy                           0.82        11
   macro avg       0.80      0.88      0.80        11
weighted avg       0.89      0.82      0.83        11
```

In [101]:

```
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
dtc.score(x_train,y_train)
preddtc=dtc.predict(x_test)
print(accuracy_score(y_test,preddtc))
print(confusion_matrix(y_test,preddtc))
print(classification_report(y_test,preddtc))
```

```
0.9090909090909091
[[7 1]
 [0 3]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       0.75      1.00      0.86         3
```

```
    accuracy                           0.91        11
   macro avg       0.88      0.94      0.90        11
weighted avg       0.93      0.91      0.91        11
```

In [102]:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
knn.score(x_train,y_train)
predknn=knn.predict(x_test)
print(accuracy_score(y_test,predknn))
print(confusion_matrix(y_test,predknn))
print(classification_report(y_test,predknn))
```

```
0.9090909090909091
[[7 1]
 [0 3]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       0.75      1.00      0.86         3

    accuracy                           0.91        11
   macro avg       0.88      0.94      0.90        11
weighted avg       0.93      0.91      0.91        11
```

In [103]:

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
rf.score(x_train,y_train)
predrf=rf.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
0.9090909090909091
[[7 1]
 [0 3]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       0.75      1.00      0.86         3

    accuracy                           0.91        11
   macro avg       0.88      0.94      0.90        11
weighted avg       0.93      0.91      0.91        11
```

In [104]:

```
from sklearn.ensemble import AdaBoostClassifier
ad=AdaBoostClassifier()
ad.fit(x_train,y_train)
ad.score(x_train,y_train)
predad=ad.predict(x_test)
print(accuracy_score(y_test,predad))
print(confusion_matrix(y_test,predad))
print(classification_report(y_test,predad))
```

```
0.9090909090909091
[[7 1]
 [0 3]]
              precision    recall  f1-score   support

           0       1.00      0.88      0.93         8
           1       0.75      1.00      0.86         3
```

```
    accuracy                           0.91        11
   macro avg       0.88      0.94      0.90        11
weighted avg       0.93      0.91      0.91        11
```

In [105]:

```
import joblib
joblib.dump(ad,'salary.pkl')
```

Out[105]:

```
['salary.pkl']
```

In [ ]:

```

```