In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
from sklearn.datasets import load_breast_cancer
df=load_breast_cancer()
```

In [3]:
```
df.data
```

Out[3]:
```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]])
```

In [4]:
```
df.feature_names
```

Out[4]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [5]:
```
df1=pd.DataFrame(df.data,columns=df.feature_names)
df1
```

Out[5]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809
...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

569 rows × 30 columns

In [6]:
```
df1["target"]=df.target
df1
```

Out[6]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809
...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

569 rows × 31 columns

In [7]:
```
df.target
```

Out[7]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

In [8]:
```
df.target_names
```

Out[8]:
```
array(['malignant', 'benign'], dtype='<U9')
```

In [9]:
```
df1= pd.DataFrame(np.c_[df.data, df.target],
columns=[list(df.feature_names)+['target']])
df1
```

Out[9]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809
...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

569 rows × 31 columns

In [10]:
```
df1.head()
```

Out[10]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 31 columns

In [11]:
```
df1.shape
```

Out[11]:
```
(569, 31)
```

In [12]:
```
x = df1.iloc[:, 0:-1]
y = df1.iloc[:, -1]
```

In [13]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_sta
```

In [14]:
```
print('Shape of X_train = ', x_train.shape)
print('Shape of y_train = ', y_train.shape)
print('Shape of X_test = ', x_test.shape)
print('Shape of y_test = ', y_test.shape)
```

```
Shape of X_train =  (455, 30)
Shape of y_train =  (455,)
Shape of X_test =  (114, 30)
Shape of y_test =  (114,)
```

In [15]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [16]:
```
classifier = RandomForestClassifier(n_estimators=100, criterion='gini')
```

In [17]:
```
classifier.fit(x_train, y_train)
```

Out[17]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [18]:
```
classifier.score(x_test, y_test)
```

Out[18]:
```
0.9385964912280702
```

In [19]:
```
classifier = RandomForestClassifier(n_estimators=100, criterion='entropy')
```

In [20]:
```
classifier.fit(x_train, y_train)
```

Out[20]:
```
RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

In [21]:
```
classifier.score(x_test, y_test)
```

Out[21]:
```
0.9298245614035088
```

In [22]:
```
patient1 = [17.99,
10.38,
122.8,
1001.0,
0.1184,
0.2776,
0.3001,
0.1471,
0.2419,
0.07871,
1.095,
0.9053,
8.589,
153.4,
0.006399,
0.04904,
0.05373,
0.01587,
0.03003,
0.006193,
25.38,
17.33,
184.6,
2019.0,
0.1622,
0.6656,
0.7119,
0.2654,
0.4601,
0.1189]
```

In [23]:
```
patient1 = np.array([patient1])
patient1
```

Out[23]:
```
array([[1.799e+01, 1.038e+01, 1.228e+02, 1.001e+03, 1.184e-01, 2.776e-01,
        3.001e-01, 1.471e-01, 2.419e-01, 7.871e-02, 1.095e+00, 9.053e-01,
        8.589e+00, 1.534e+02, 6.399e-03, 4.904e-02, 5.373e-02, 1.587e-02,
        3.003e-02, 6.193e-03, 2.538e+01, 1.733e+01, 1.846e+02, 2.019e+03,
        1.622e-01, 6.656e-01, 7.119e-01, 2.654e-01, 4.601e-01, 1.189e-01]])
```

In [24]:
```
classifier.predict(patient1)
```

Out[24]:
```
array([0.])
```

In [25]:
```
df.target_names
```

Out[25]:
```
array(['malignant', 'benign'], dtype='<U9')
```

In [26]:
```
pred = classifier.predict(patient1)
```

In [27]:
```
if pred[0] == 0:
    print('Patient has Cancer (malignant tumor)')
else:
    print('Patient has no Cancer ( benign)')
```

```
Patient has Cancer (malignant tumor)
```

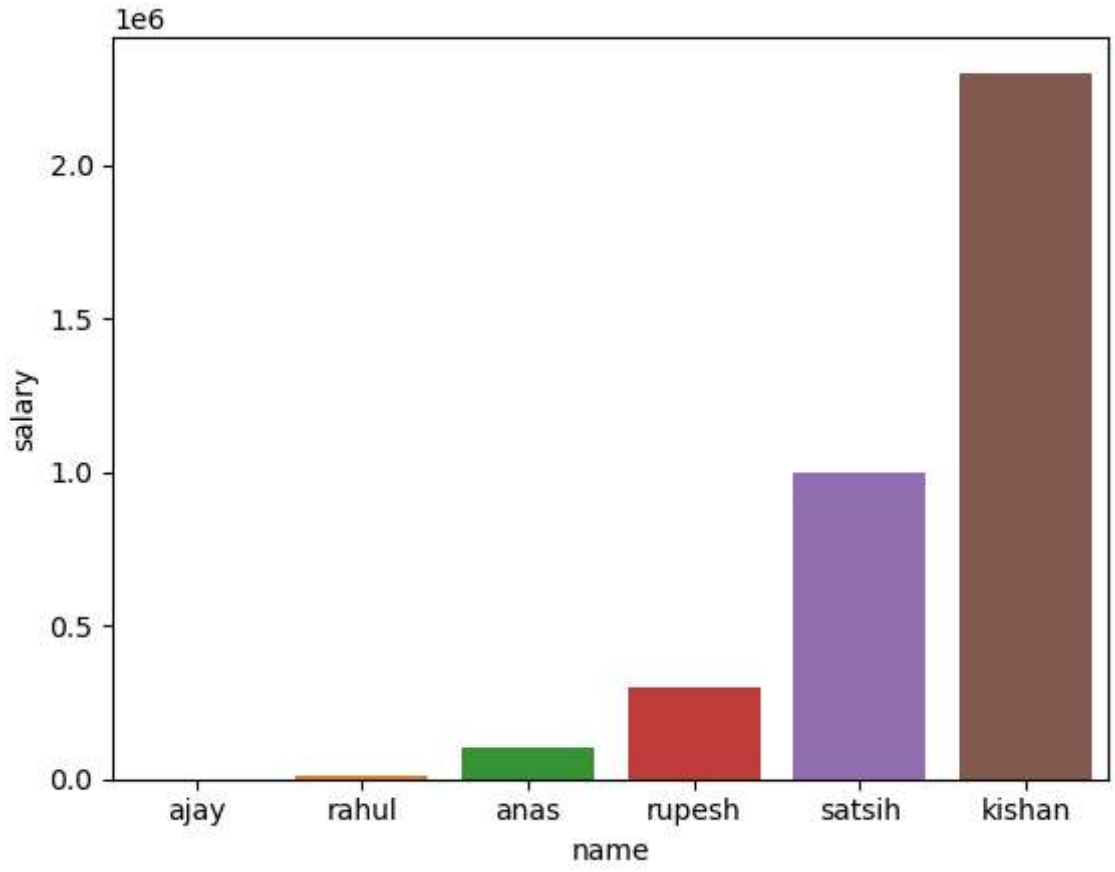
In [28]:
```
data=pd.DataFrame({"name":["ajay","rahul","anas","rupesh","satsih","kishan"],
                   "salary":[1000,10000,100000,300000,1000000,2300000]})
data
```

Out[28]:

	name	salary
0	ajay	1000
1	rahul	10000
2	anas	100000
3	rupesh	300000
4	satsih	1000000
5	kishan	2300000

In [29]:
```
import seaborn as sns
sns.barplot(x="name",y="salary",data=data)
```

Out[29]:
```
<Axes: xlabel='name', ylabel='salary'>
```



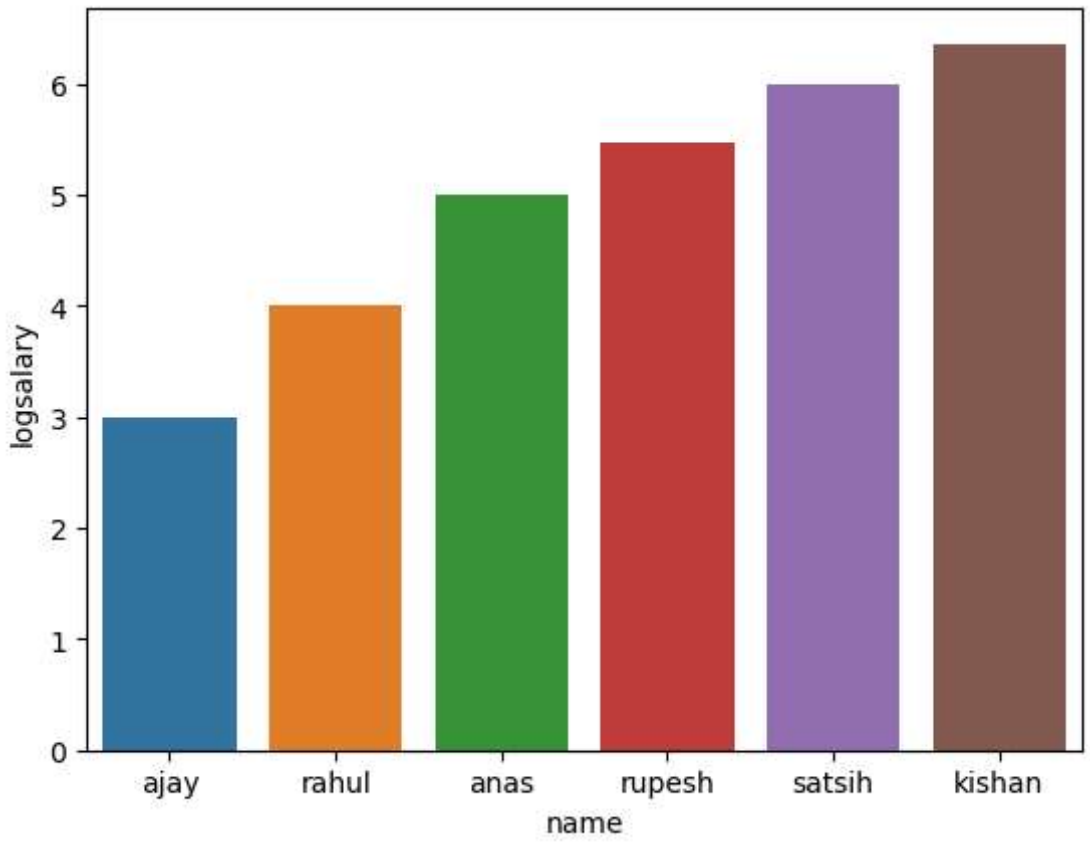
In [30]:
```
from math import log10
data["logsalary"]=data.salary.apply(lambda i:log10(i))
data
```

Out[30]:

	name	salary	logsalary
0	ajay	1000	3.000000
1	rahul	10000	4.000000
2	anas	100000	5.000000
3	rupesh	300000	5.477121
4	satsih	1000000	6.000000
5	kishan	2300000	6.361728

In [31]:
```
import seaborn as sns
sns.barplot(x="name",y="logsalary",data=data)
```

Out[31]:
```
<Axes: xlabel='name', ylabel='logsalary'>
```



In [32]:
```
data
```

Out[32]:

	name	salary	logsalary
0	ajay	1000	3.000000
1	rahul	10000	4.000000
2	anas	100000	5.000000
3	rupesh	300000	5.477121
4	satsih	1000000	6.000000
5	kishan	2300000	6.361728

In [33]:
```
max=data.salary.max()
max
```

Out[33]:
```
2300000
```

In [34]:
```
min= data.salary.min()
min
```

Out[34]:
```
1000
```

In [35]:
```
pd.set_option("display.precision",2)
```

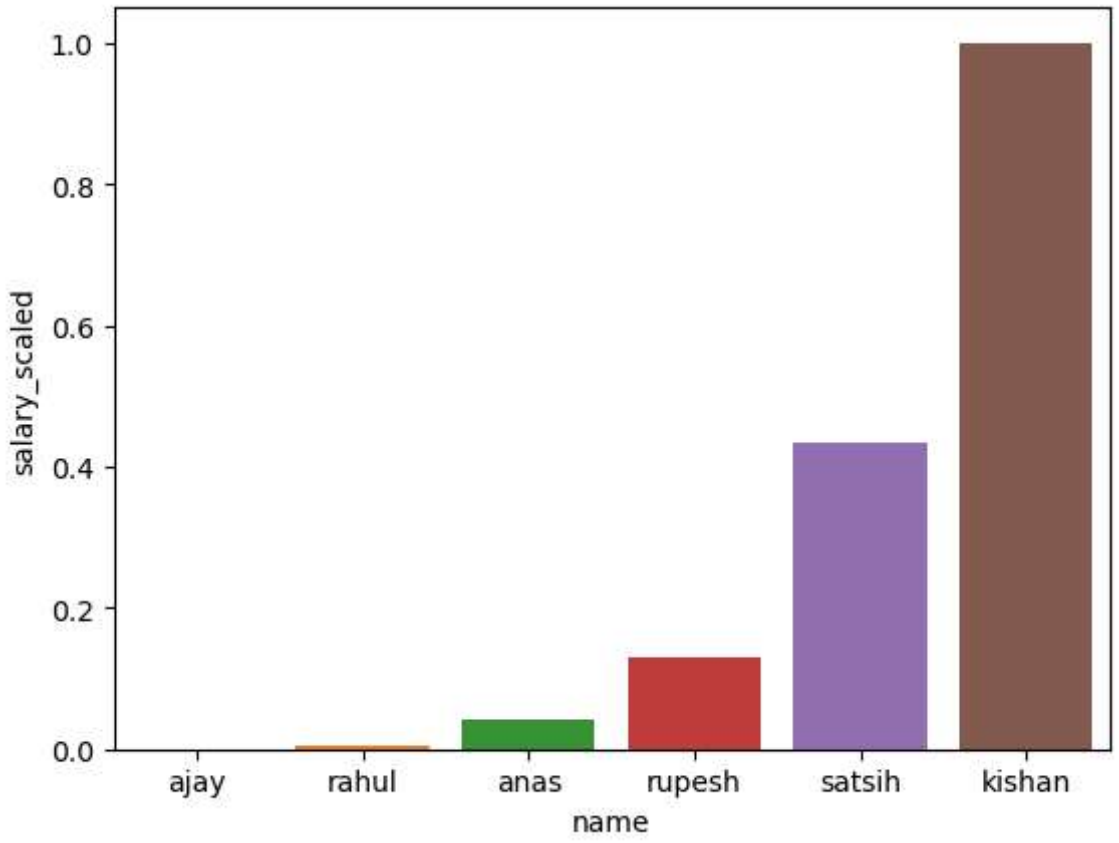
In [36]:
```
data["salary_scaled"]=data.salary.apply(lambda i:(i-min)/(max-min))
data
```

Out[36]:

	name	salary	logsalary	salary_scaled
0	ajay	1000	3.00	0.00e+00
1	rahul	10000	4.00	3.91e-03
2	anas	100000	5.00	4.31e-02
3	rupesh	300000	5.48	1.30e-01
4	satsih	1000000	6.00	4.35e-01
5	kishan	2300000	6.36	1.00e+00

In [37]:
```
import seaborn as sns
sns.barplot(x="name",y="salary_scaled",data=data)
```

Out[37]:
```
<Axes: xlabel='name', ylabel='salary_scaled'>
```



In [38]:
```
mean=data.salary.mean()
mean
```

Out[38]:
```
618500.0
```

In [39]:
```
std=data.salary.std()
std
```

Out[39]:
```
904969.3364970993
```

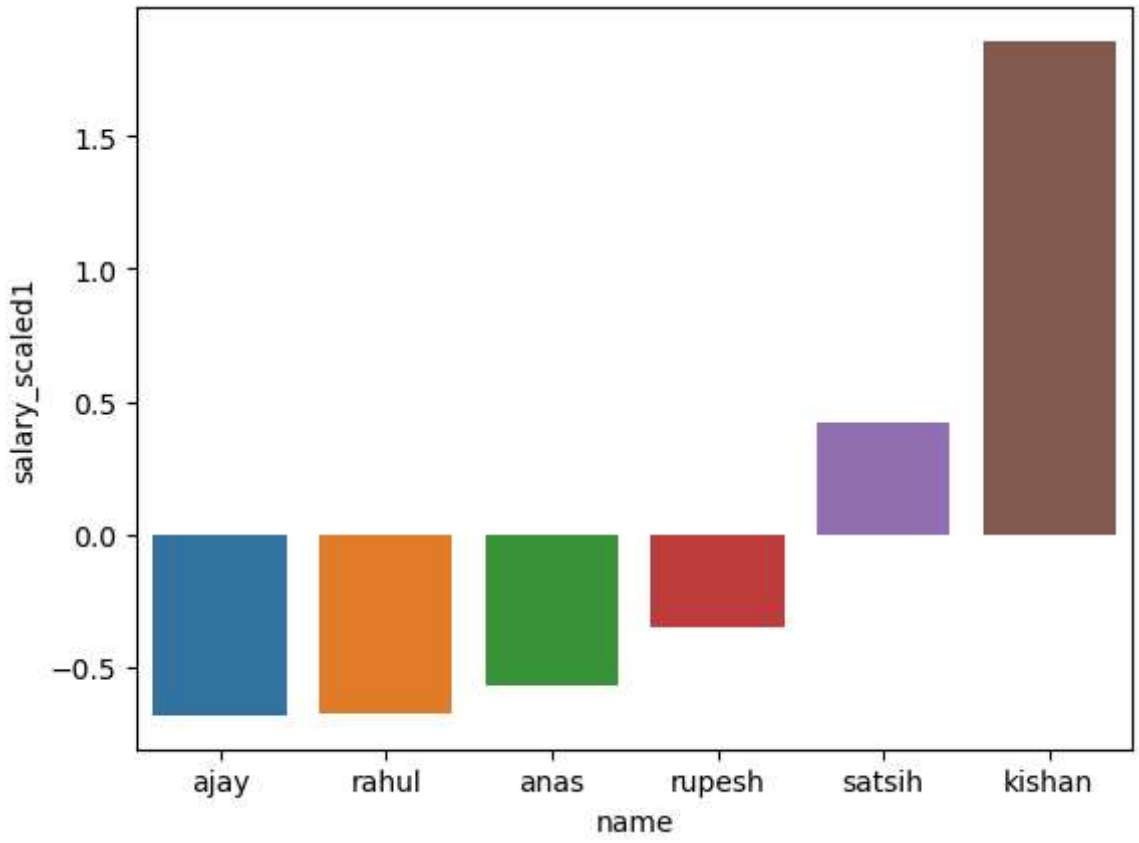
In [40]:
```
data["salary_scaled1"]=data.salary.apply(lambda i:(i-mean)/std)
data
```

Out[40]:

	name	salary	logsalary	salary_scaled	salary_scaled1
0	ajay	1000	3.00	0.00e+00	-0.68
1	rahul	10000	4.00	3.91e-03	-0.67
2	anas	100000	5.00	4.31e-02	-0.57
3	rupesh	300000	5.48	1.30e-01	-0.35
4	satsih	1000000	6.00	4.35e-01	0.42
5	kishan	2300000	6.36	1.00e+00	1.86

In [41]:
```
import seaborn as sns
sns.barplot(x="name",y="salary_scaled1",data=data)
```

Out[41]:
```
<Axes: xlabel='name', ylabel='salary_scaled1'>
```



In [42]:
```
# confusion matrix
```

In [43]:
```
y_test
```

Out[43]:
```
169    1.0
137    1.0
188    1.0
443    1.0
345    1.0
      ...
538    1.0
123    1.0
224    1.0
115    1.0
425    1.0
Name: (target,), Length: 114, dtype: float64
```

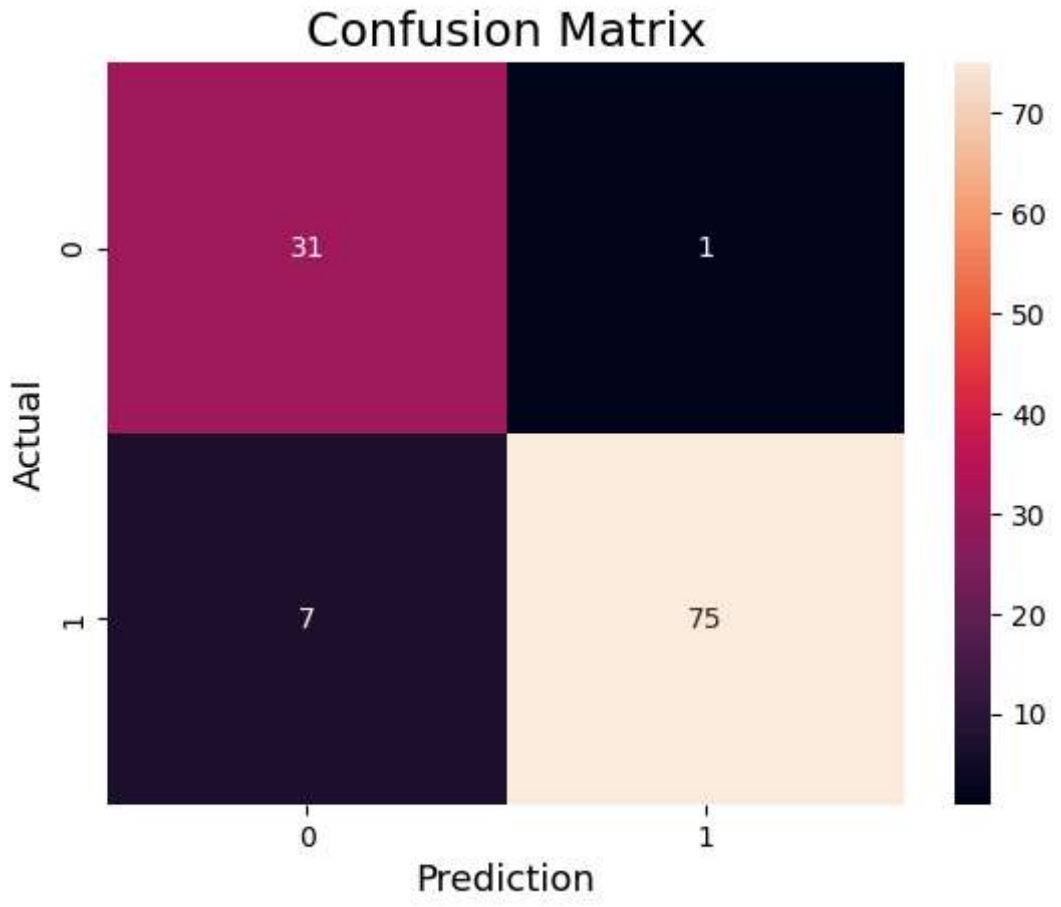
In [44]:
```
y_predicted=classifier.predict(x_test)
y_predicted
```

Out[44]:
```
array([1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
       1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 0., 0., 1., 1., 1.,
       1., 0., 1., 0., 1., 0., 0., 1., 1., 1., 1., 1., 0., 1., 1., 0., 1.,
       1., 1., 1., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 1., 0., 0.,
       1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [45]:
```
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
cm = confusion_matrix(y_predicted,y_test)
cm
```

Out[45]:
```
array([[31,  1],
       [ 7, 75]], dtype=int64)
```

In [46]:
```
sns.heatmap(cm,
            annot=True,
            fmt='g',
            xticklabels=['0','1'],
            yticklabels=['0','1'])
plt.xlabel('Prediction',fontsize=13)
plt.ylabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```



In [47]:
```
tp=75
tn=33
fn=5
fp=1
```

In [48]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

In [49]:
```
accuracy = accuracy_score(y_predicted,y_test)
print("Accuracy   :", accuracy)
```

```
Accuracy   : 0.9298245614035088
```

In [50]:
```
ac=(tp+tn)/(tp+tn+fp+fn)
ac
```

Out[50]:
```
0.9473684210526315
```

In [51]:
```
precision =precision_score(y_predicted,y_test)
print("Precision   :", precision)
```

```
Precision   : 0.9868421052631579
```

In [52]:
```
pr=tp/(tp+fp)
pr
```

Out[52]:
```
0.9868421052631579
```

In [53]:
```
recall =recall_score(y_predicted,y_test)
print("recall   :", recall)
```

```
recall   : 0.9146341463414634
```

In [54]:
```
rc=tp/(tp+fn)
rc
```

Out[54]:
```
0.9375
```

In [55]:
```
f1_score=f1_score(y_predicted,y_test)
print("f1_score  :",f1_score)
```

```
f1_score  : 0.949367088607595
```

In [56]:
```
f1_sc=2*(pr*rc)/(pr+rc)
f1_sc
```

Out[56]:
```
0.9615384615384615
```

In [57]:
```
error=1-ac
error
```

Out[57]:
```
0.052631578947368474
```

In [58]:
```
df1
```

Out[58]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.12	0.28	0.30	0.15	0.24
1	20.57	17.77	132.90	1326.0	0.08	0.08	0.09	0.07	0.18
2	19.69	21.25	130.00	1203.0	0.11	0.16	0.20	0.13	0.21
3	11.42	20.38	77.58	386.1	0.14	0.28	0.24	0.11	0.26
4	20.29	14.34	135.10	1297.0	0.10	0.13	0.20	0.10	0.18
...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11	0.12	0.24	0.14	0.17
565	20.13	28.25	131.20	1261.0	0.10	0.10	0.14	0.10	0.18
566	16.60	28.08	108.30	858.1	0.08	0.10	0.09	0.05	0.16
567	20.60	29.33	140.10	1265.0	0.12	0.28	0.35	0.15	0.24
568	7.76	24.54	47.92	181.0	0.05	0.04	0.00	0.00	0.16

569 rows × 31 columns

In [59]:
```
df1.columns
```

Out[59]:
```
MultiIndex([(            'mean radius',),
            (           'mean texture',),
            (         'mean perimeter',),
            (              'mean area',),
            (        'mean smoothness',),
            (       'mean compactness',),
            (         'mean concavity',),
            (    'mean concave points',),
            (          'mean symmetry',),
            ( 'mean fractal dimension',),
            (           'radius error',),
            (          'texture error',),
            (        'perimeter error',),
            (             'area error',),
            (       'smoothness error',),
            (      'compactness error',),
            (        'concavity error',),
            (   'concave points error',),
            (         'symmetry error',),
            ('fractal dimension error',),
            (           'worst radius',),
            (          'worst texture',),
            (        'worst perimeter',),
            (             'worst area',),
            (       'worst smoothness',),
            (      'worst compactness',),
            (        'worst concavity',),
            (   'worst concave points',),
            (         'worst symmetry',),
            ('worst fractal dimension',),
            (                'target',)],
           )
```

In [60]:
```
df2=pd.DataFrame(df1.columns,columns=["item"])
df2
```

Out[60]:

	item
0	(mean radius,)
1	(mean texture,)
2	(mean perimeter,)
3	(mean area,)
4	(mean smoothness,)
5	(mean compactness,)
6	(mean concavity,)
7	(mean concave points,)
8	(mean symmetry,)
9	(mean fractal dimension,)
10	(radius error,)
11	(texture error,)
12	(perimeter error,)
13	(area error,)
14	(smoothness error,)
15	(compactness error,)
16	(concavity error,)
17	(concave points error,)
18	(symmetry error,)
19	(fractal dimension error,)
20	(worst radius,)
21	(worst texture,)
22	(worst perimeter,)
23	(worst area,)
24	(worst smoothness,)
25	(worst compactness,)
26	(worst concavity,)
27	(worst concave points,)
28	(worst symmetry,)
29	(worst fractal dimension,)
30	(target,)

In [ ]:

In [ ]:

In [ ]: