In [ ]:
```
Aim: To implement logistic regression for logistic data
```

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score
```

In [2]:
```
dataset=pd.read_csv("logistic_data.csv")
```

In [3]:
```
dataset
```

Out[3]:

	4.5192	2.6487	1.0
0	2.4443	1.5438	1.0
1	4.2409	1.8990	1.0
2	5.8097	2.4711	1.0
3	6.4423	3.3590	1.0
4	5.8097	3.2406	1.0
...	...	...	...
94	5.9868	7.3641	0.0
95	4.6711	6.2592	0.0
96	7.5810	8.3703	0.0
97	4.6457	8.5676	0.0
98	4.6457	8.1676	0.0

99 rows × 3 columns

In [4]:
```
x=dataset.iloc[:,[0,1]].values
x
```

Out[4]:
```
array([[2.4443, 1.5438],
       [4.2409, 1.899 ],
       [5.8097, 2.4711],
       [6.4423, 3.359 ],
       [5.8097, 3.2406],
       [6.3917, 3.8128],
       [6.8725, 4.4441],
       [6.7966, 3.6747],
       [8.163 , 4.7401],
       [7.4038, 3.8917],
       [7.6316, 4.602 ],
       [7.7581, 5.7265],
       [6.5688, 4.9571],
       [5.3543, 3.9903],
       [4.4686, 3.0236],
       [2.9757, 2.0568],
       [2.4443, 1.2676],
       [0.9008, 1.169 ],
       [2.1154, 1.7411],
       [3.2794, 1.386 ],
       [4.165 , 1.5636],
       [4.8482, 1.8793],
       [3.33  , 2.7868],
       [5.1518, 3.5563],
       [6.2652, 4.0693],
       [6.2652, 4.3849],
       [7.2014, 1.5438],
       [7.6569, 2.412 ],
       [6.1387, 1.7806],
       [4.4939, 1.4057],
       [4.8735, 2.6093],
       [5.5314, 3.0828],
       [6.0121, 3.9311],
       [7.1508, 4.7598],
       [7.7075, 5.3122],
       [8.3148, 5.7068],
       [8.5172, 5.1149],
       [8.7449, 5.4109],
       [7.8593, 3.8128],
       [6.999 , 3.2406],
       [5.5061, 2.9052],
```

```
       [4.9241, 2.6882],
       [6.6447, 3.8325],
       [7.6822, 4.5428],
       [8.0364, 5.7857],
       [8.9221, 6.5552],
       [7.8593, 5.253 ],
       [6.5941, 5.2333],
       [6.0374, 4.7598],
       [2.7227, 4.5822],
       [1.9383, 3.6549],
       [1.6852, 2.9841],
       [4.3168, 4.4244],
       [3.4312, 3.7536],
       [5.4808, 5.2728],
       [4.1144, 4.8387],
       [3.2034, 4.4244],
       [4.1144, 5.3911],
       [5.1012, 6.0817],
       [4.8988, 5.5687],
       [5.9615, 6.4565],
       [5.7591, 6.0028],
       [6.6953, 6.7722],
       [5.7338, 6.6538],
       [6.6194, 7.1471],
       [7.2014, 7.5219],
       [7.2014, 6.8314],
       [8.5931, 7.6206],
       [7.7581, 7.1865],
       [7.7581, 7.7784],
       [5.1012, 7.6009],
       [4.2156, 6.496 ],
       [3.4818, 5.8055],
       [2.3684, 5.0163],
       [1.7864, 4.1876],
       [0.9008, 3.4379],
       [0.9008, 5.7857],
       [1.9636, 6.3382],
       [1.4069, 4.9571],
       [2.419 , 6.8511],
       [2.8745, 6.0817],
       [4.0132, 7.1668],
       [4.6711, 7.226 ],
```

```
       [5.1771, 8.1533],
       [6.2146, 7.4825],
       [5.4555, 7.0484],
       [5.9868, 8.5084],
       [4.0891, 7.5417],
       [2.3937, 7.2063],
       [1.331 , 6.5355],
       [1.7358, 5.4503],
       [2.4443, 5.8449],
       [3.1781, 4.8979],
       [4.6711, 5.8055],
       [5.9868, 7.3641],
       [4.6711, 6.2592],
       [7.581 , 8.3703],
       [4.6457, 8.5676],
       [4.6457, 8.1676]])
```

In [6]:
```
y=dataset.iloc[:,2].values
y
```

Out[6]:
```
array([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., 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., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [11]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=0)
```

In [13]:
```
xtrain
```

Out[13]:
```
array([[6.0374, 4.7598],
       [6.8725, 4.4441],
       [4.6457, 8.1676],
       [4.6711, 7.226 ],
       [0.9008, 5.7857],
       [5.9615, 6.4565],
       [2.8745, 6.0817],
       [1.331 , 6.5355],
       [7.7581, 7.1865],
       [1.6852, 2.9841],
       [7.6569, 2.412 ],
       [2.1154, 1.7411],
       [3.2034, 4.4244],
       [5.7338, 6.6538],
       [1.7864, 4.1876],
       [4.2409, 1.899 ],
       [5.7591, 6.0028],
       [6.6447, 3.8325],
       [4.9241, 2.6882],
       [5.8097, 3.2406],
       [2.9757, 2.0568],
       [0.9008, 1.169 ],
       [5.5061, 2.9052],
       [7.8593, 3.8128],
       [6.3917, 3.8128],
       [1.7358, 5.4503],
       [4.8988, 5.5687],
       [2.4443, 1.5438],
       [7.7075, 5.3122],
       [6.1387, 1.7806],
       [1.9383, 3.6549],
       [7.7581, 5.7265],
       [8.3148, 5.7068],
       [5.1518, 3.5563],
       [4.3168, 4.4244],
       [7.6316, 4.602 ],
       [5.5314, 3.0828],
       [7.2014, 6.8314],
       [4.1144, 5.3911],
       [2.419 , 6.8511],
       [5.4555, 7.0484],
```

```
[6.0121, 3.9311],
[6.2146, 7.4825],
[4.4686, 3.0236],
[2.3937, 7.2063],
[3.2794, 1.386 ],
[4.4939, 1.4057],
[2.7227, 4.5822],
[7.581 , 8.3703],
[4.6457, 8.5676],
[7.7581, 7.7784],
[4.165 , 1.5636],
[4.6711, 5.8055],
[3.4818, 5.8055],
[1.9636, 6.3382],
[6.2652, 4.3849],
[8.7449, 5.4109],
[4.0132, 7.1668],
[7.8593, 5.253 ],
[6.999 , 3.2406],
[7.2014, 7.5219],
[5.1012, 6.0817],
[6.5688, 4.9571],
[5.1012, 7.6009],
[4.0891, 7.5417],
[8.5172, 5.1149],
[4.8482, 1.8793],
[5.1771, 8.1533],
[7.4038, 3.8917],
[4.6711, 6.2592],
[8.5931, 7.6206],
[6.6194, 7.1471],
[6.5941, 5.2333],
[8.0364, 5.7857]])
```

In [15]:
```
xtest
```

Out[15]:
```
array([[7.2014, 1.5438],
       [5.9868, 8.5084],
       [5.8097, 2.4711],
       [4.1144, 4.8387],
       [0.9008, 3.4379],
       [3.1781, 4.8979],
       [2.4443, 1.2676],
       [2.3684, 5.0163],
       [5.4808, 5.2728],
       [5.9868, 7.3641],
       [3.4312, 3.7536],
       [2.4443, 5.8449],
       [1.4069, 4.9571],
       [5.3543, 3.9903],
       [6.7966, 3.6747],
       [4.8735, 2.6093],
       [3.33  , 2.7868],
       [6.2652, 4.0693],
       [7.1508, 4.7598],
       [8.163 , 4.7401],
       [7.6822, 4.5428],
       [6.6953, 6.7722],
       [6.4423, 3.359 ],
       [4.2156, 6.496 ],
       [8.9221, 6.5552]])
```

In [16]:
```
ytrain
```

Out[16]:
```
array([1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 1., 0.,
       1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 1., 1., 0., 1., 1., 1.,
       0., 1., 1., 0., 0., 0., 0., 1., 0., 1., 0., 1., 1., 0., 0., 0., 0.,
       1., 0., 0., 0., 1., 1., 0., 1., 1., 0., 0., 1., 0., 0., 1., 1., 0.,
       1., 0., 0., 0., 1., 1.])
```

In [17]:
```
ytest
```

Out[17]:
```
array([1., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1.,
       1., 1., 1., 1., 0., 1., 0., 1.])
```

In [19]:
```
classifier=LogisticRegression(random_state=0)
classifier.fit(xtrain,ytrain)
```

Out[19]:
```
LogisticRegression(random_state=0)
```
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 [20]:
```
classifier.classes_
```

Out[20]:
```
array([0., 1.])
```

In [21]:
```
classifier.intercept_
```

Out[21]:
```
array([3.12787746])
```

In [22]:
```
classifier.coef_
```

Out[22]:
```
array([[ 1.51533124, -2.31207756]])
```

In [23]:
```
classifier.predict_proba(xtest)
```

Out[23]:
```
array([[2.83459374e-05, 9.99971654e-01],
       [9.99431658e-01, 5.68342304e-04],
       [1.98901946e-03, 9.98010981e-01],
       [8.61152257e-01, 1.38847743e-01],
       [9.69403746e-01, 3.05962545e-02],
       [9.67091846e-01, 3.29081542e-02],
       [1.98206335e-02, 9.80179367e-01],
       [9.92469887e-01, 7.53011257e-03],
       [6.80923186e-01, 3.19076814e-01],
       [9.92049261e-01, 7.95073949e-03],
       [5.86940624e-01, 4.13059376e-01],
       [9.98748375e-01, 1.25162458e-03],
       [9.97977513e-01, 2.02248726e-03],
       [1.17581738e-01, 8.82418262e-01],
       [7.16905650e-03, 9.92830944e-01],
       [1.12070682e-02, 9.88792932e-01],
       [1.50506663e-01, 8.49493337e-01],
       [3.86722347e-02, 9.61327765e-01],
       [4.93278086e-02, 9.50672191e-01],
       [1.05811093e-02, 9.89418891e-01],
       [1.38482802e-02, 9.86151720e-01],
       [9.15628182e-01, 8.43718176e-02],
       [5.91796336e-03, 9.94082037e-01],
       [9.95943411e-01, 4.05658894e-03],
       [1.83669179e-01, 8.16330821e-01]])
```

In [24]:
```
y_pred=classifier.predict(xtest)
print(y_pred)
```

```
[1. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0.
 1.]
```

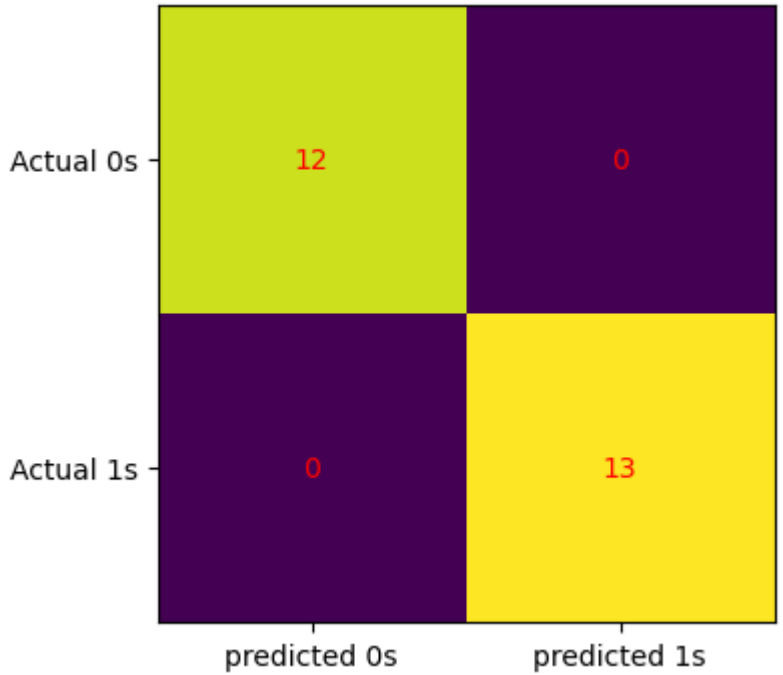
In [26]:
```
print("Accuracy:",accuracy_score(ytest,y_pred))
```

```
Accuracy: 1.0
```

In [27]:
```
cm=confusion_matrix(ytest,y_pred)
print("confusion matrix:\n",cm)
```

```
confusion matrix:
 [[12  0]
 [ 0 13]]
```

In [29]:
```
fig,ax=plt.subplots(figsize=(4,4))
ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0,1),ticklabels=("predicted 0s","predicted 1s"))
ax.yaxis.set(ticks=(0,1),ticklabels=("Actual 0s","Actual 1s"))
ax.set_ylim(1.5,-0.5)
for i in range(2):
    for j in range(2):
        ax.text(j,i,cm[i,j],ha="center",va="center",color="red")
plt.show()
```



In [ ]: