## https://github.com/Abhishek841428/ETE-18SCSE1010334.git

In [1]:

```
#2.Write a program to demonstrate the working of the Logistic Regression.
#Use an appropriate data set the implementation.

print('Abhishek Kumar 18SCSE1010334')
```

```
Abhishek Kumar 18SCSE1010334
```

In [2]:

```
#example is related to a single-variate binary classification problem.
#This is the most straightforward kind of classification problem.

import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

In [3]:

```
x = np.arange(10).reshape(-1, 1)
y = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
```

In [4]:

```
x
```

Out[4]:

```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5],
       [6],
       [7],
       [8],
       [9]])
```

In [5]:

```
y
```

Out[5]:

```
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
```

In [6]:

```
model = LogisticRegression(solver='liblinear', random_state=0)
```

In [7]:

```
model.fit(x, y)
```

Out[7]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='liblinear', tol=0.0001, ver
bose=0,
                   warm_start=False)
```

In [8]:

```
model = LogisticRegression(solver='liblinear', random_state=0).fit(x, y)
```

In [9]:

```
model.classes_
```

Out[9]:

```
array([0, 1])
```

In [10]:

```
model.intercept_
```

Out[10]:

```
array([-1.04608067])
```

In [11]:

```
model.coef_
```

Out[11]:

```
array([[0.51491375]])
```

In [12]:

```
model.predict_proba(x)
```

Out[12]:

```
array([[0.74002157, 0.25997843],
       [0.62975524, 0.37024476],
       [0.5040632 , 0.4959368 ],
       [0.37785549, 0.62214451],
       [0.26628093, 0.73371907],
       [0.17821501, 0.82178499],
       [0.11472079, 0.88527921],
       [0.07186982, 0.92813018],
       [0.04422513, 0.95577487],
       [0.02690569, 0.97309431]])
```

In [13]:

```
model.predict(x)
```

Out[13]:

```
array([0, 0, 0, 1, 1, 1, 1, 1, 1, 1])
```

In [14]:

```
model.score(x, y)
```

Out[14]:

```
0.9
```

In [15]:

```
confusion_matrix(y, model.predict(x))
```

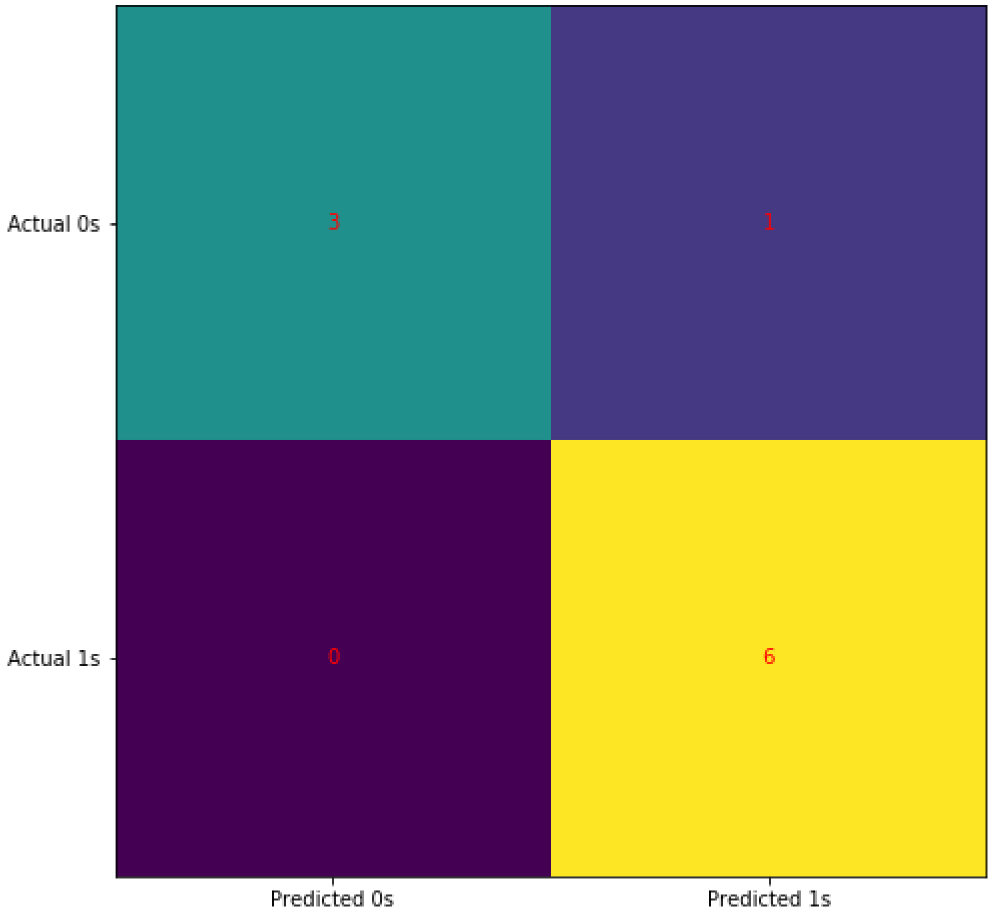
Out[15]:

```
array([[3, 1],
       [0, 6]], dtype=int64)
```

In [16]:

```
cm = confusion_matrix(y, model.predict(x))

fig, ax = plt.subplots(figsize=(8, 8))
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 [17]:

```
print(classification_report(y, model.predict(x)))
```

```
              precision    recall  f1-score   support

           0       1.00      0.75      0.86         4
           1       0.86      1.00      0.92         6

    accuracy                           0.90        10
   macro avg       0.93      0.88      0.89        10
weighted avg       0.91      0.90      0.90        10
```

In [18]:

```
model = LogisticRegression(solver='liblinear', C=10.0, random_state=0)
model.fit(x, y)
```

Out[18]:

```
LogisticRegression(C=10.0, class_weight=None, dual=False, fit_intercep
t=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='liblinear', tol=0.0001, ver
bose=0,
                   warm_start=False)
```

In [19]:

```
model.intercept_
```

Out[19]:

```
array([-3.51335372])
```

In [20]:

```
model.coef_
```

Out[20]:

```
array([[1.12066084]])
```

In [21]:

```
model.predict_proba(x)
```

Out[21]:

```
array([[0.97106534, 0.02893466],
       [0.9162684 , 0.0837316 ],
       [0.7810904 , 0.2189096 ],
       [0.53777071, 0.46222929],
       [0.27502212, 0.72497788],
       [0.11007743, 0.88992257],
       [0.03876835, 0.96123165],
       [0.01298011, 0.98701989],
       [0.0042697 , 0.9957303 ],
       [0.00139621, 0.99860379]])
```

In [22]:

```
model.predict(x)
```

Out[22]:

```
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
```

In [23]:

```
model.score(x, y)
```

Out[23]:

```
1.0
```

In [24]:

```
confusion_matrix(y, model.predict(x))
```

Out[24]:

```
array([[4, 0],
       [0, 6]], dtype=int64)
```

In [25]:

```
print(classification_report(y, model.predict(x)))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         6

    accuracy                           1.00        10
   macro avg       1.00      1.00      1.00        10
weighted avg       1.00      1.00      1.00        10
```

In [ ]:

```

```