In [1]:
```
#Aim: to implement logistic regression for iris data set
```

In [2]:
```
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 [3]:
```
dataset=pd.read_csv("iris.csv")
```

In [4]:
```
dataset
```

Out[4]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Virginica
146	6.3	2.5	5.0	1.9	Virginica
147	6.5	3.0	5.2	2.0	Virginica
148	6.2	3.4	5.4	2.3	Virginica
149	5.9	3.0	5.1	1.8	Virginica

150 rows × 5 columns

In [5]:
```
x=dataset.iloc[:,[0,1,2,3]].values
y=dataset.iloc[:,4].values
```

In [6]:
```
x
```

Out[6]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3]
```

```
In [7]: y
```

```
Out[7]: array(['Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa',
               'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa',
               'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa',
               'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa',
               'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa',
               'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa',
               'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa',
               'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa', 'Setosa',
               'Setosa', 'Setosa', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Versicolor',
               'Versicolor', 'Versicolor', 'Versicolor', 'Virginica', 'Virginica',
               'Virginica', 'Virginica', 'Virginica', 'Virginica', 'Virginica',
               'Virginica', 'Virginica', 'Virginica', 'Virginica', 'Virginica',
               'Virginica', 'Virginica', 'Virginica', 'Virginica', 'Virginica',
               'Virginica', 'Virginica', 'Virginica', 'Virginica', 'Virginica',
               'Virginica', 'Virginica', 'Virginica', 'Virginica', 'Virginica',
               'Virginica', 'Virginica', 'Virginica', 'Virginica', 'Virginica',
               'Virginica', 'Virginica', 'Virginica', 'Virginica', 'Virginica',
               'Virginica', 'Virginica', 'Virginica', 'Virginica', 'Virginica',
               'Virginica', 'Virginica', 'Virginica', 'Virginica', 'Virginica',
               'Virginica', 'Virginica', 'Virginica'], dtype=object)
```

```
In [8]: xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=0)
```

In [9]:
```
xtrain
```

Out[9]:
```
array([[5.9, 3. , 4.2, 1.5],
       [5.8, 2.6, 4. , 1.2],
       [6.8, 3. , 5.5, 2.1],
       [4.7, 3.2, 1.3, 0.2],
       [6.9, 3.1, 5.1, 2.3],
       [5. , 3.5, 1.6, 0.6],
       [5.4, 3.7, 1.5, 0.2],
       [5. , 2. , 3.5, 1. ],
       [6.5, 3. , 5.5, 1.8],
       [6.7, 3.3, 5.7, 2.5],
       [6. , 2.2, 5. , 1.5],
       [6.7, 2.5, 5.8, 1.8],
       [5.6, 2.5, 3.9, 1.1],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.3, 4.7, 1.6],
       [5.5, 2.4, 3.8, 1.1],
       [6.3, 2.7, 4.9, 1.8],
       [6.3, 2.8, 5.1, 1.5],
       [4.9, 2.5, 4.5, 1.7],
       [6.3, 2.5, 5. , 1.9]
```

In [10]:
```
classifier=LogisticRegression(random_state=43)
classifier.fit(xtrain,ytrain)
```

```
/Users/rahul/anaconda3/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:460: 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 (https://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#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[10]:
```
▾ LogisticRegression
LogisticRegression(random_state=43)
```

```
In [11]: classifier.classes_
```

```
Out[11]: array(['Setosa', 'Versicolor', 'Virginica'], dtype=object)
```

```
In [12]: classifier.intercept_
```

```
Out[12]: array([  9.25389453,   1.75982611, -11.01372064])
```

```
In [13]: classifier.coef_
```

```
Out[13]: array([[-0.41737243,  0.85016164, -2.33197673, -0.98816398],
                [ 0.52060574, -0.29765625, -0.22056126, -0.7110104 ],
                [-0.10323331, -0.55250539,  2.55253799,  1.69917438]])
```

In [14]:

```
classifier.predict_proba(xtest)
```

Out[14]:

```
array([[1.17923827e-04, 5.61477126e-02, 9.43734364e-01],
       [1.26289274e-02, 9.60454578e-01, 2.69164949e-02],
       [9.84397656e-01, 1.56023051e-02, 3.85623800e-08],
       [1.25178024e-06, 2.31525672e-02, 9.76846181e-01],
       [9.70234825e-01, 2.97650128e-02, 1.62601257e-07],
       [2.01667884e-06, 5.94451237e-03, 9.94053471e-01],
       [9.81899513e-01, 1.81004166e-02, 7.04438513e-08],
       [2.84241321e-03, 7.47090500e-01, 2.50067087e-01],
       [1.50915530e-03, 7.38523100e-01, 2.59967745e-01],
       [2.05288164e-02, 9.35891370e-01, 4.35798137e-02],
       [9.22423042e-05, 1.59473395e-01, 8.40434363e-01],
       [6.98627884e-03, 8.09990600e-01, 1.83023122e-01],
       [4.08220464e-03, 7.93602339e-01, 2.02315456e-01],
       [3.05681770e-03, 7.60910322e-01, 2.36032861e-01],
       [3.87699722e-03, 7.10277101e-01, 2.85845902e-01],
       [9.82815600e-01, 1.71843437e-02, 5.65458427e-08],
       [6.72901453e-03, 7.56465847e-01, 2.36805138e-01],
       [1.14291867e-02, 8.45110735e-01, 1.43460078e-01],
       [9.67582194e-01, 3.24175913e-02, 2.14237353e-07],
       [9.82872113e-01, 1.71278272e-02, 5.96878608e-08],
       [8.34495449e-04, 1.93259567e-01, 8.05905937e-01],
       [1.03255905e-02, 7.11148279e-01, 2.78526130e-01],
       [9.44128885e-01, 5.58700663e-02, 1.04838226e-06],
       [9.75498569e-01, 2.45012638e-02, 1.67521226e-07],
       [1.36907259e-03, 4.26371225e-01, 5.72259702e-01],
       [9.94203372e-01, 5.79661840e-03, 9.65289787e-09],
       [9.50240522e-01, 4.97583459e-02, 1.13240457e-06],
       [1.07122659e-02, 9.00995202e-01, 8.82925322e-02],
       [1.40885249e-01, 8.52873823e-01, 6.24092794e-03],
       [9.61492012e-01, 3.85075385e-02, 4.49514538e-07],
       [9.90728441e-05, 1.15644174e-01, 8.84256753e-01],
       [1.19870263e-02, 6.84360565e-01, 3.03652408e-01],
       [9.68058486e-01, 3.19413643e-02, 1.50147241e-07],
       [1.28526268e-03, 3.57780651e-01, 6.40934086e-01],
       [1.48834296e-05, 3.38270057e-02, 9.66158111e-01],
       [4.81305475e-02, 8.80739722e-01, 7.11297308e-02],
       [9.44629269e-01, 5.53703395e-02, 3.91123233e-07],
       [6.02622733e-04, 3.11031121e-01, 6.88366257e-01]])
```

In [15]:

```
y_pred=classifier.predict(xtest)
print(y_pred)
```

```
['Virginica' 'Versicolor' 'Setosa' 'Virginica' 'Setosa' 'Virginica'
 'Setosa' 'Versicolor' 'Versicolor' 'Versicolor' 'Virginica' 'Versicolor'
 'Versicolor' 'Versicolor' 'Versicolor' 'Setosa' 'Versicolor' 'Versicolor'
 'Setosa' 'Setosa' 'Virginica' 'Versicolor' 'Setosa' 'Setosa' 'Virginica'
 'Setosa' 'Setosa' 'Versicolor' 'Versicolor' 'Setosa' 'Virginica'
 'Versicolor' 'Setosa' 'Virginica' 'Virginica' 'Versicolor' 'Setosa'
 'Virginica']
```

In [16]:

```
print("Accuracy:",accuracy_score(ytest,y_pred))
```

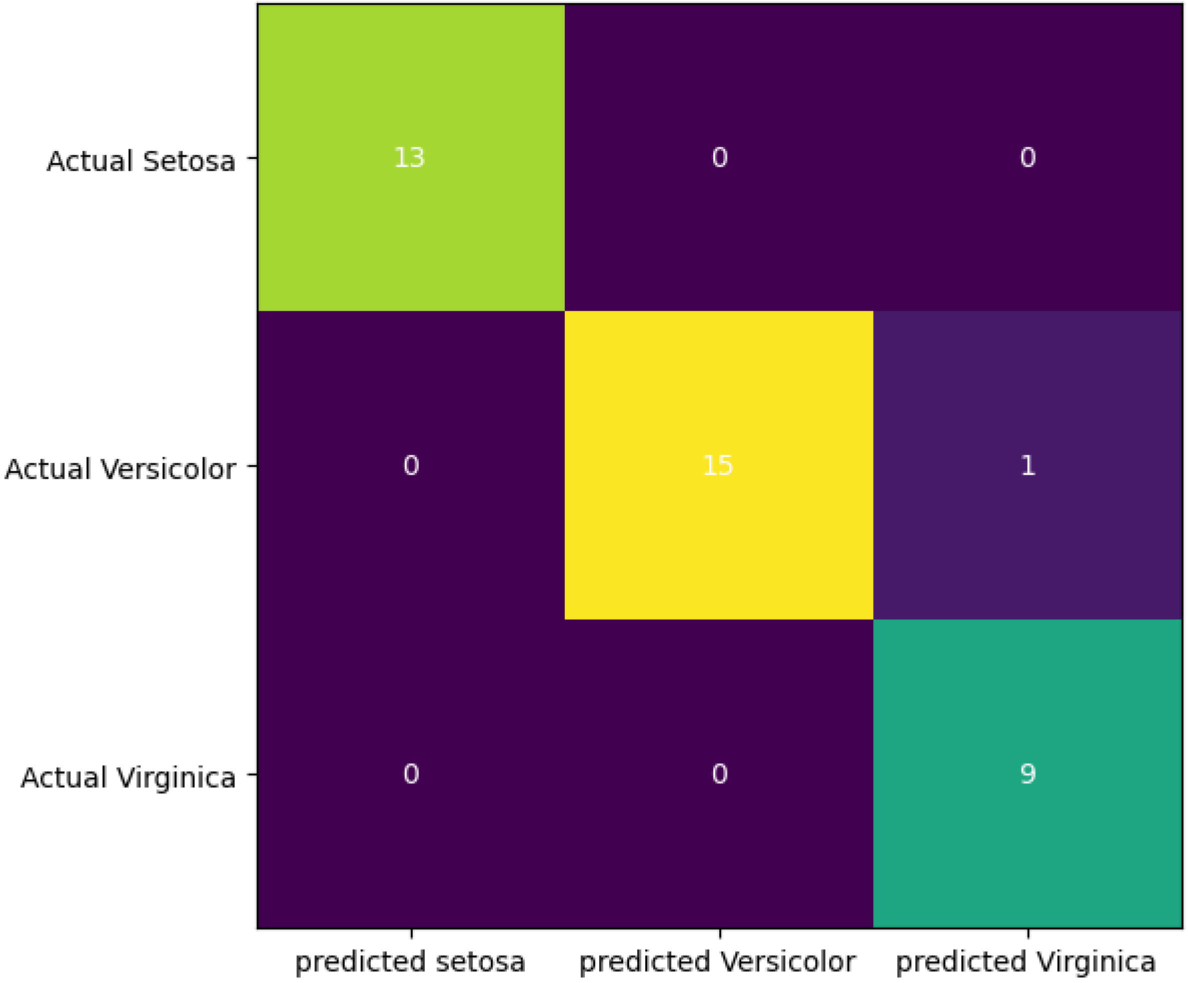
```
Accuracy: 0.9736842105263158
```

In [17]:

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

```
confusion matrix:
 [[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]
```

In [18]:
```
fig,ax=plt.subplots(figsize=(6,6))
ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0,1,2),ticklabels=("predicted setosa","predicted Versicolor","predicted Virginica"))
ax.yaxis.set(ticks=(0,1,2),ticklabels=("Actual Setosa","Actual Versicolor","Actual Virginica"))
ax.set_ylim(2.5,-0.5)
for i in range(3):
    for j in range(3):
        ax.text(j,i,cm[i,j],ha="center",va="center",color="white")
plt.show()
```


Actual Setosa
Actual Versicolor
Actual Virginica
predicted setosa
predicted Versicolor
predicted Virginica