```
########################################
# TWO-class logistic regression
#X[0:NEXP][0:Nfeatures]
#T[0,NEXP]
# sklearn solver
########################################
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

#==================================
M       = 100
NCLASS      = 2
NEXP        = NCLASS*M
NFEATURES   = 2

# NEXP NUMBER OF EXPERIMENTS
# NFEATURES number of features
# X[0:NEXP-1][0:NFEATURES-1]
# TRAINING VECTOR
# T[0,NEXP-1]
# NCLASS number of classes

X  = np.zeros((NEXP,NFEATURES))
T  = np.zeros((NEXP))

#==================================
#DATA GENERATOR
cov1 = [[.1,0],[0,.1]]
mean1 =[0,0]
cov2 = [[.1,0],[0,.1]]
mean2 = [2,2]
XRAW1 = np.random.multivariate_normal(mean1, cov1,M)
XRAW2 = np.random.multivariate_normal(mean2, cov2,M)
#==================================
for i in range(0,M,1):
    X[i,0]=XRAW1[i,0]
    X[i,1]=XRAW1[i,1]
    T[i]=0

    X[i+M,0]=XRAW2[i,0]
    X[i+M,1]=XRAW2[i,1]
    T[i+M]=1

plt.scatter(X[0:M ,0],X[0:M ,1],alpha=0.8,lw=3,s=3,label='0')
plt.scatter(X[M:NEXP,0],X[M:NEXP,1],alpha=0.8,lw=3,s=3,label='1')
plt.legend()
plt.show()


#================================================
```

```
# Perform standard scaling on X->XSCALED
#================================================
mean  = np.zeros((NFEATURES))
var   = np.zeros((NFEATURES))
std   = np.zeros((NFEATURES))
for j in range (0,NFEATURES,1):
    mean[j]=np.mean(X[:,j])
    var [j]=np.var (X[:,j])
    std [j]=np.sqrt (var[j])

XSCALED= np.zeros((NEXP,NFEATURES))
for j in range (0,NFEATURES,1):
    XSCALED[:,j]= (X[:,j] -mean[j])/std[j]
#------------------------------------------------




#================================================
# Solver
#   L(limited memory)-BFGS solver
#   C= reciprocal regularization magnitude
#================================================
clf= LogisticRegression(C=1e9,solver='lbfgs')
model = clf.fit(XSCALED,T)
#result
W0 = clf.intercept_
W  = clf.coef_   #row-wise order
print(' the model')
print('W shape:',np.shape(W));
print('W0=',W0);
print('W =',W);

#================================================
# classification and error
#================================================
# classification using training data
Posterior= np.zeros((NCLASS))
error_count=0

for i in range (0,NEXP,1):
    tmp= W0+ np.dot(W[0][:],XSCALED[i,:])
    Posterior[0] = 1/(1+np.exp(tmp)) #PROB(CLASS=0)
    Posterior[1] = 1-Posterior[0]    #PROB(CLASS=1)
    if(np.argmax(Posterior)!= T[i]):
        error_count=error_count+1
print( 'error:', error_count,'of',NEXP)
#================================================

exit();
```