```
########################################
#Three-class logistic regression
#2 features: X[0:NEXP-1][0:NFEATURES-1]
#3 classes: targets:  T[0,NEXP-1]= {0,1,2}
# sklearn solver
########################################
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

#==================================
M       = 100
NCLASS      = 3
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]
cov3 = [[.1,0],[0,.1]]
mean3 = [-2,-2]

XRAW1 = np.random.multivariate_normal(mean1, cov1,M)
XRAW2 = np.random.multivariate_normal(mean2, cov2,M)
XRAW3 = np.random.multivariate_normal(mean3, cov3,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

    X[i+2*M,0]=XRAW3[i,0]
    X[i+2*M,1]=XRAW3[i,1]
    T[i+2*M]=2
```

T(0) = 0
T(1) = 1
T(2) = 2

```
plt.scatter(X[0:M ,0],X[0:M ,1],alpha=0.8,lw=3,s=3,label='0')
plt.scatter(X[M:2*M,0],X[M:2*M,1],alpha=0.8,lw=3,s=3,label='1')
plt.scatter(X[2*M:NEXP,0],X[2*M:NEXP,1],alpha=0.8,lw=3,s=3,label='2')
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
#================================================
#multi-class logistic regression
clf= LogisticRegression(C=1e9,multi_class="multinomial",solver='lbfgs')

model = clf.fit(XSCALED,T)

#result
W0 = clf.intercept_
W  = clf.coef_   #row-wise order
print(' the model',model)
print('W shape:',np.shape(W));
print('W0=',W0);
print('W =',W);

#================================================
# classification and error count
#ALT to
#clf.predict(XSCALED[i,:]))
#clf.predict_proba(XSCALED[i,:]))
#================================================

error_count=0
Y  = np.zeros((NEXP,NCLASS))
ERROR  = np.zeros((NEXP))
Posterior  = np.zeros((NCLASS))

#POSTERIOR PROB
```

```
for i in range (0,NEXP,1):

    for j in range(0,NCLASS,1):
        Y[i,j]= W0[j] + np.dot(W[j][:],XSCALED[i,:])

    for k in range (0,NCLASS,1):
        Posterior[k]=0
        for j in range (0,NCLASS,1):
            Posterior[k]=Posterior[k]+ np.exp(Y[i,j]-Y[i,k])
        Posterior[k]=1/Posterior[k]
    if(np.argmax(Posterior) != T[i]):
        error_count=error_count +1
print( 'error using posterior prob:', error_count,'of',NEXP)

exit()
```