In [1]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from math import *

import matplotlib.pyplot as plt

from sklearn import linear_model
from sklearn.metrics import log_loss
import math
```

Without sklearn

In [123]:
```
def grad():
    X, Y = make_classification(n_samples= 50000 , n_features=7, n_infor
mative=6, n_redundant=1,
                               n_classes=2, weights=[0.7], class_sep=0.7, r
andom_state=42)
    X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.3,st
ratify =Y,random_state=42)


    df=pd.DataFrame(X_train)
    df['y']=y_train
    df=df.values
    w_po=np.zeros(X.shape[1]).reshape(-1,1)##initilizing weight


    b_po=np.array([[0]])#initilizing intercept

    eta0  = 0.0001#learning rate
    alpha = 0.0001#hyperparameter term in reg
```

```
    train=[]
    test=[]
    #log-loss befor SGD for train and test

    print('Befor_SGD :-')
    s_tr=[]
    for i in X_train:
        sig=1/(1+np.exp(-(np.dot(w_po.T,i.reshape(-1,1))+b_po)))

        s_tr.append(sig[0][0])
    c=0
    for i in range(len(y_train)):
        c=c+(-((y_train[i]*math.log10(s_tr[i]))+(1-y_train[i])*math.log
10(1-s_tr[i])))
    tr_loss=(c/len(y_train))
    print('train_log_loss_befor_SGD  : ',tr_loss)

    s_te=[]
    for i in X_test:
        sig=1/(1+np.exp(-(np.dot(w_po.T,i.reshape(-1,1))+b_po)))

        s_te.append(sig[0][0])
    d=0
    for i in range(len(y_test)):
        d=d+(-((y_test[i]*math.log10(s_te[i]))+(1-y_test[i])*math.log10
(1-s_te[i])))
    te_loss=(d/len(y_test))
    print('test_log_loss_befor_SGD  : ',te_loss)



    epoch=12
    for po in range(epoch):

        for j in range(len(X_train)):
            rand_val=[np.random.randint(0,len(y_train))]#generating one
 random interget
            #selecting randomly one point x & y
            x=df[rand_val,:-1]
```

```
            y=df[rand_val,-1]
            #sigmoid function
            sig=1/(1+np.exp(-(np.dot(w_po.T,x.reshape(-1,1))+b_po)))
            #updating weight
            w_po = (1-(alpha*eta0))*w_po + (alpha*x.reshape(-1,1))*(y-s
ig)
            #updating intercept
            b_po = (1-(alpha*eta0))*b_po + alpha*(y-sig)
        #finding Log-loss for training value
        s_train=[]
        for i in X_train:
            sig=1/(1+np.exp(-(np.dot(w_po.T,i.reshape(-1,1))+b_po)))

            s_train.append(sig[0][0])
        c=0
        for i in range(len(y_train)):
            c=c+(-((y_train[i]*math.log10(s_train[i]))+(1-y_train[i])*m
ath.log10(1-s_train[i])))
        train_loss=(c/len(y_train))
        train.append(train_loss)
        #finding Log-loss for testing value
        s_test=[]
        for i in X_test:
            sig=1/(1+np.exp(-(np.dot(w_po.T,i.reshape(-1,1))+b_po)))

            s_test.append(sig[0][0])
        d=0
        for i in range(len(y_test)):
            d=d+(-((y_test[i]*math.log10(s_test[i]))+(1-y_test[i])*math
.log10(1-s_test[i])))
        test_loss=(d/len(y_test))
        test.append(test_loss)



    posa=[]
    print('\nFinial Report :- \n')
    print('1. weight : ',w_po.T[0])
    print('2. intercept : ',b_po[0])
```

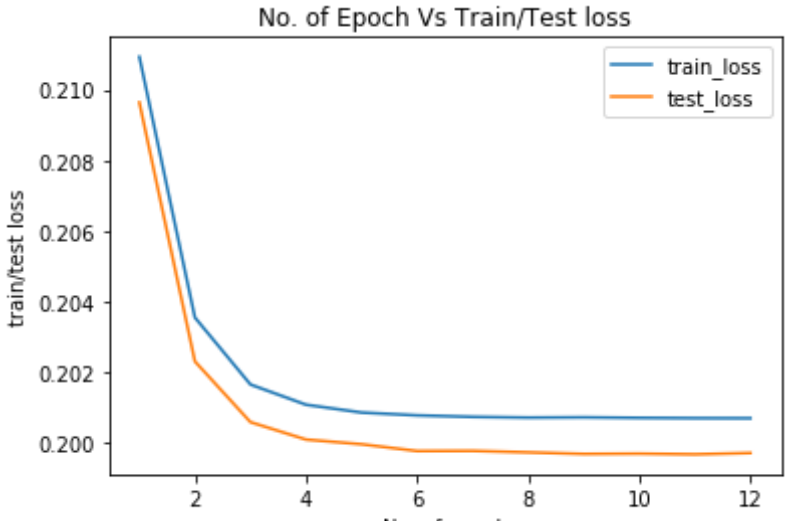
```
    print('3. train loss :',train_loss)
    print('4. test loss :',test_loss)
    plt.title('No. of Epoch Vs Train/Test loss')
    plt.xlabel('No. of epoch')
    plt.ylabel('train/test loss')
    plt.plot(range(1,epoch+1),train)
    plt.plot(range(1,epoch+1),test)
    plt.legend(('train_loss','test_loss'))


grad()
```

```
Befor_SGD :-
train_log_loss_befor_SGD  :  0.30102999566405614
test_log_loss_befor_SGD  :  0.3010299956640423

Finial Report :-

1. weight :  [ 0.67610029 -0.15035462  0.32956211 -0.00386871 -0.261987
46  0.40911314
  0.55523696]
2. intercept :  [-0.64254213]
3. train loss : 0.2006746695304598
4. test loss : 0.19968787866133156
```



No. of epoch

In [ ]:

With sklearn

In [114]:
```
X, Y = make_classification(n_samples= 50000 , n_features=7, n_informati
ve=6, n_redundant=1,
                           n_classes=2, weights=[0.7], class_sep=0.7, r
andom_state=42)
```

In [115]:
```
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=.3,stratif
y =Y,random_state=42)
```

In [ ]:

In [116]:
```
clf = linear_model.SGDClassifier(eta0=0.0001, alpha=0.0001, loss='log',
 penalty='l2', tol=1e-4, verbose=2, learning_rate='constant')
```

In [117]:
```
clf.fit(X_train, y_train)
```

```
-- Epoch 1
Norm: 0.66, NNZs: 7, Bias: -0.335513, T: 35000, Avg. loss: 0.536325
Total training time: 0.01 seconds.
-- Epoch 2
Norm: 0.85, NNZs: 7, Bias: -0.457297, T: 70000, Avg. loss: 0.475254
Total training time: 0.02 seconds.
-- Epoch 3
Norm: 0.94, NNZs: 7, Bias: -0.525910, T: 105000, Avg. loss: 0.466194
Total training time: 0.03 seconds.
-- Epoch 4
Norm: 0.99, NNZs: 7, Bias: -0.565090, T: 140000, Avg. loss: 0.463633
Total training time: 0.04 seconds.
-- Epoch 5
Norm: 1.02, NNZs: 7, Bias: -0.590896, T: 175000, Avg. loss: 0.462733
```

```
Total training time: 0.05 seconds.
-- Epoch 6
Norm: 1.04, NNZs: 7, Bias: -0.610440, T: 210000, Avg. loss: 0.462371
Total training time: 0.05 seconds.
-- Epoch 7
Norm: 1.05, NNZs: 7, Bias: -0.622577, T: 245000, Avg. loss: 0.462262
Total training time: 0.06 seconds.
-- Epoch 8
Norm: 1.06, NNZs: 7, Bias: -0.632037, T: 280000, Avg. loss: 0.462183
Total training time: 0.06 seconds.
-- Epoch 9
Norm: 1.06, NNZs: 7, Bias: -0.639766, T: 315000, Avg. loss: 0.462173
Total training time: 0.07 seconds.
-- Epoch 10
Norm: 1.07, NNZs: 7, Bias: -0.644575, T: 350000, Avg. loss: 0.462148
Total training time: 0.07 seconds.
-- Epoch 11
Norm: 1.06, NNZs: 7, Bias: -0.645110, T: 385000, Avg. loss: 0.462141
Total training time: 0.07 seconds.
-- Epoch 12
Norm: 1.06, NNZs: 7, Bias: -0.648651, T: 420000, Avg. loss: 0.462133
Total training time: 0.09 seconds.
Convergence after 12 epochs took 0.09 seconds
```

Out[117]:
```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0001,
              fit_intercept=True, l1_ratio=0.15, learning_rate='constan
t',
              loss='log', max_iter=1000, n_iter_no_change=5, n_jobs=Non
e,
              penalty='l2', power_t=0.5, random_state=None, shuffle=Tru
e,
              tol=0.0001, validation_fraction=0.1, verbose=2, warm_star
t=False)
```

In [118]:
```
print('Weight : ',clf.coef_[0] )
print('Intercept : ',clf.intercept_)
```

```
Weight :  [ 0.66631866 -0.14484723  0.3285386  -0.00314992 -0.26216377
```

```
0.4120236
  0.56812363]
Intercept :  [-0.64865053]
```

Difference

In [132]:
```
a=np.array([ 0.66631866, -0.14484723  ,0.3285386 , -0.00314992, -0.2621
6377,  0.4120236, 0.56812363])
b=np.array([ 0.67610029 ,-0.15035462 , 0.32956211, -0.00386871, -0.2619
8746  ,0.40911314,0.55523696])
c=-0.64865053
d=-0.64254213
print('Weight difference : ',abs(a-b))
print('intercept difference : ',abs(c-d))
```

```
Weight difference :  [0.00978163 0.00550739 0.00102351 0.00071879 0.000
17631 0.00291046
 0.01288667]
intercept difference :  0.006108399999999903
```

In [ ]:
```

```