在 [ ]中：
```
import torch
a = torch.cuda.is_available()
```

实验一 参考代码

在 [ ]中：
```
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings("ignore")
# 这个方法只是解决了表面，没有根治
```

在 [ ]中：

```
cancer=load_breast_cancer()#加载乳腺癌数据
X_train,X_test,y_train,y_test=train_test_split(cancer.data,cancer.target,test_size=0.2)
model = LogisticRegression()
model.fit(X_train,y_train)
train_score=model.score(X_train,y_train)
test_score=model.score(X_test,y_test)
print(f'train score:{train_score:.6f};testscore:{test_score:.6f}')

y_pred=model.predict(X_test)
accuracy_score_value= accuracy_score(y_test,y_pred)
recall_score_value = recall_score(y_test,y_pred)
precision_score_value=precision_score(y_test,y_pred)
classification_report_value=classification_report(y_test,y_pred)
print(f"准确率：{accuracy_score_value}")
print(f"召回率：{recall_score_value}")
print(f"精确率:{precision_score_value}")
print(classification_report_value)
```

```
train score:0.962637;testscore:0.929825
准确率：0.9298245614035088
召回率：0.9705882352941176
精确率:0.9166666666666666
              precision    recall  f1-score   support

           0       0.95      0.87      0.91        46
           1       0.92      0.97      0.94        68

    accuracy                           0.93       114
   macro avg       0.93      0.92      0.93       114
weighted avg       0.93      0.93      0.93       114
```

实验二参考代码

在 [ ]中:
```
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

在 [ ]中:
```
def loadTrainData():
    cancer=load_breast_cancer()#加载乳腺癌数据
    X=cancer.data
    #加载乳腺癌判别特征
    y=cancer.target#两个TAG,y=时为阴性， y=1时为阳性
    #将数据集划分为训练集和测试集，测试集占比为8,2
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
    X_train = X_train.T
    X_test = X_test.T
    return X_train,X_test,y_train,y_test
```

在 [ ]中:
```
def sigmoid(inx):
    from numpy import exp
    return 1.0/(1.0 + exp(-inx))
```

初始化参数

在 [ ]中:
```
def initialize_para(dim):
    mu = 0
    sigma =0.1
    np.random.seed()
    w = np.random.normal(mu,sigma,dim)
    w = np.reshape(w,(dim,1))
    b=0
    return w,b
```

前向传播

在 [ ]中：

```
def propagate(w,b,X,Y):
    #eps防止log运算遇到e
    eps = 1e-5
    m = X.shape[1]
    #计算初步运算结果
    A =sigmoid(np.dot(w.T,x)+b)
    #计算损失函数值大小
    cost =-1 / m*np.sum(np.multiply(Y,np.log(A+eps))+np.multiply(1-Y,np.log(1 - A + eps)))
    #计算梯度值
    dw = 1 /m*np.dot(X,(A -Y).T)
    db = 1 /m*np.sum(A-Y)
    cost = np.squeeze(cost)

    grads ={"dw":dw,
        "db":db}
    #返回损失函数大小以及反向传播的梯度值
    return grads,cost,A
```

num1 terations梯度下降次数

learning_rate学习率

在 [ ]中：

```
def optimize(w,b,X,Y,num_iterations,learning_rate):
    costs=[] #记录损失函数值
    #循环进行梯度下降
    for i in range(num_iterations):
        print(i)
        grads,cost,pre_y = propagate(w,b,X,Y)
        dw = grads["dw"]
        db = grads["db"]
        w=w - learning_rate * dw
        b=b - learning_rate * db
        #每100次循环记录一次损失函数大小并打印
        if i%100==0:
            costs.append(cost)
        if i%100==0:
            pre_Y[pre_Y >=0.5]=1
            pre_Y[pre_Y< 0.5]=0
            pre_Y=pre_Y.astype(np.int)
            acc =1 - np.sum(pre_Y^Y)/len(Y)
            print("Iteration:{} Loss ={},Acc = {}".format(i,cost,acc))
    #最终参数值
    params = {"w":w,
        "b":b}
    return params,costs
```

在 [ ]中：

```
def predict(w,b,X):
    #样本个数
    m = X.shape[1]
    #初始化预测输出
    Y_prediction = np.zeros((1,m))
    #转置参数向量w
    w=w.reshape(X.shape[0],1)
    #预测结果
    Y_hat = sigmoid(np.dot(w.T,X)+b)
    #将结果按照0.5的阈值转化为6/1
    for i in range(Y_hat.shape[1]):
        if Y_hat[:,i]>0.5:
            Y_prediction[:,i]=1
        else:
            Y_prediction[:,i]=0
    return Y_prediction
```

在 [ ]中：

```
#训川练以及预测
def Logisticmodel(X_train,Y_train,X_test,Y_test,num_iterations=1000,learning_rate=0.1):
    #初始化参数w,b
    w,b = initialize_para(X_train.shape[0])
    #梯度下降找到最优参数
    parameters,costs = optimize(w,b,X_train,Y_train,num_iterations,learning_rate)
    w = parameters["w"]
    b = parameters["b"]
    #训练集测试集的预测结果
    Y_prediction_train = predict(w,b,X_train)
    Y_prediction_test = predict(w,b,X_test)
    Y_prediction_test = Y_prediction_test.T
    #模型评价
    accuracy_score_value = accuracy_score(Y_test,Y_prediction_test)
    recall_score_value = recall_score(Y_test,Y_prediction_test)
    precision_score_value = precision_score(Y_test,Y_prediction_test)
    classification_report_value = classification_report(Y_test,Y_prediction_test)
    print("准确率：",accuracy_score_value)
    print("召回率：",recall_score_value)
    print("精确率：",precision_score_value)
    print(classification_report_value)
    d = {"costs":costs,
        "Y_prediction_test":Y_prediction_test,
        "Y_prediction_train":Y_prediction_train,
        "w":w,"b":b,
        "learning_rate":learning_rate,
        "num_iterations":num_iterations}
    return d
```

在 [ ]中：

```
if __name__='_main_':
    X_train,X_test,y_train,y_test = loadTrainData()
    Logisticmodel(X_train,y_train,X_test,y_test)
```