```
#%% md
Question 1
#%%
import numpy as np
#%%
def generateData(d, N, noise=0.1):
    np.random.seed(12345)
    X = np.concatenate([np.ones([N, 1]), np.random.randn(N, d)], axis=1)
    lambdaTrue = -1 + 2 * np.tan(np.random.randn(len(X[0])))
    Y = 1 * ((X @ lambdaTrue + noise * np.random.randn(N)) > 0)
    D = X, Y
    return D
#%%
def split(D, train_size):
    x, y = D
    x_trainning = []
    x_testing = []
    y_traning = []
    y_testing = []
    for i in range(0, train_size):
        x_trainning.append(x[i])
        y_traning.append(y[i])
    for i in range(train_size, len(y)):
        x_testing.append(x[i])
        y_testing.append(y[i])
    return x_trainning, x_testing, y_traning, y_testing
#%%
def model(x, par):
    z = x @ par
    # return 1 / (np.exp(-z)+1)
    return np.power(1 / (1 + np.power(np.e, -z)), -1)
#%%
def dModel(x, par):
    f = np.power((1 + np.power(np.e, (np.dot(x, par)))), -1)
    gradient = np.dot(f * (1 - f), x)
```

```python
    return gradient
#%%
def objective(par, data):
    ell = 0
    X, Y = data
    for n in range(0, len(X)):
        x, y = X[n], Y[n]
        # f = np.power((np.dot(par,x)),np.e)
        f = model(X, par)
        if y == 1:
            f = (f)
        else:
            deltat = (1 - f)
        s = -(np.log(f))
        ell = ell + s
    return ell
#%%
def gradient(par, data):
    grad = np.zeros(len(par))
    X, Y = data
    for n in range(0, len(X)):
        f = model(X[n], par)
        x, y = X[n], Y[n]
        # deltaf = (dModel(X,par))
        s = -(y - f) * (dModel(x, par)) / (y * f + (1 - y) * (1 - f))
        # s = -(y-f)/(y*f+(1-y)*(1-f))
        grad = grad + s
    return grad
#%%
def train(par0, eta, T, data):
    par = par0
    obj = []
    for t in range(0, T):
        ell = objective(par, data)
        obj.append(ell)
        grad = gradient(par, data)
        par = par - grad * eta
    return par, obj
#%%
lambda1 = np.transpose(np.full(11, 0.01))
```

```
d = 10
N = 100
D = generateData(d, N)
x_trainning, x_testing, y_traning, y_testing = split(D, 50)
eta = 0.01
epoch = []
itterations = 5
# tryInit = [np.random.randn(len(X[0]))for m in range(5)]
for i in range(0, itterations):
    par, values = train(np.random.normal(0, 1, len(x_trainning[0])), eta, itterations, (x_trainning, y_traning))
    epoch.append((par, values))
#%%
def ER(par, data):
    ER = 0
    X, Y = data
    for n in range(0, len(D) - 1):
        x, y = X[n], Y[n]
        yHat = model(x, par[n])
        s = 1 * (y != yHat)
        ER = ER + s
    return ER / len(data)
#%%
ERs = []
for i in epoch:
    ERs.append(ER(i, (x_testing, y_testing)))
print(ERs)
#%% md
Question 2
#%%
import pandas as pd

#%%
data = pd.read_csv("CarsDataSetForCW1.csv",sep = ",")
Z = data[['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'year', 'origin']].to_numpy()
```

```
Y = 1 * (Z[:, 0] >= np.mean(Z[:, 0]))
X, Y = Z[:, 1 :], Y
print(Z)
#%%

D_train = (X[:len(data)//2], Y[:len(data)//2])
D_test = (X[len(data)//2:], Y[len(data)//2:])

#%%

```