Homework 2

对数几率回归的实现。

使用python以及NumPy库，建立模型，然后利用梯度下降的方法求解这个模型。

先将西瓜数据集上的数据存入data.csv中，在python文件中使用panda库读取这个csv文件，再进行处理。

data.csv文件如下：

|  |
| --- |
| 密度,含糖率,好瓜  0.697,0.460,1  0.774,0.376,1  0.634,0.264,1  0.608,0.318,1  0.556,0.215,1  0.403,0.237,1  0.481,0.149,1  0.437,0.211,1  0.666,0.091,0  0.243,0.267,0  0.245,0.057,0  0.343,0.099,0  0.639,0.161,0  0.657,0.198,0  0.360,0.370,0  0.593,0.042,0  0.719,0.103,0 |

python文件如下：

|  |
| --- |
| import pandas as pd  import numpy as np  df = pd.read\_csv("./homework2/data.csv")  X = [list(df['密度']), list(df['含糖率'])]  Y = list(df['好瓜'])  def DataTransform(X: list, Y: list):      X = np.vstack((X, np.ones((1, len(X[0]))))).T      Y = np.array(Y).T.reshape((X.shape[0], 1))      return X, Y  X, Y = DataTransform(X, Y)  def sigmoid(beta, X):      return np.sum(1 / (1 + np.exp(beta \* X)), 1).reshape((X.shape[0], 1))  def Loglikelyhood(beta, X, Y):      p0 = sigmoid(beta=beta, X=X)      return np.sum(np.log(Y \* (1 - p0) + (1 - Y) \* p0))  def Gradient(beta, X, Y):      p1 = 1 - sigmoid(beta=beta, X=X)      return -(X.T @ (Y - p1)).T  learning\_rate = 0.02  num\_iteration = 100000 # 迭代次数  beta = np.random.uniform(0, 1, size=(X.shape[1],))  for i in range(num\_iteration):      gradient = Gradient(beta, X, Y)      beta = beta - learning\_rate \* gradient      pass  maxival = -Loglikelyhood(beta=beta, X=X, Y=Y)  print(beta)  print(maxival) |

运行出来，在如下取值处得到最大的函数值

