任务	逻辑回归模型	组别
1	Machine Learning	#3

机器学习报告

小组#3

小组成员：帅灿宇 20354240

万俊麟 20354123

廖逸霖 20354079

成员贡献度：1:1:1

任务完成概况：

利用逻辑回归模型对数据建模，使用 python 进行代码编写，自定义函数进行运算得出相应的模型参数，得到最终的回归模型，计算损失函数并绘制出数据回归结果的 ROC 曲线。

1.自定义的函数：

（1）损失函数计算函数

（2）SGD 迭代函数

（3）准确率计算函数

（4）TP，FP 计算函数

（5）图像处理函数

2．数据处理结果：

（1）最终得出的 function 函数：$y = 2.3153x_1 + 1.8374x_2 + 0.8814$

（2）Final Train Loss（最终训练 loss） = 0.2031482

（3）Final Test Loss（最终测试 loss） = 0.0698756

（4）绘制出 ROC 曲线下的面积 AUC=0.98828125

目录:

1 逻辑回归模型描述：

问题：二分类任务，将线性回归模型产生的预测出的实值。

解决办法：利用广义线性回归思想，找到单调可微单数将分类任务的真实标记与线性回归模型的预测值联系起来，转化为二分类的 0/1 值。

1.1 广义线性回归：

对于给定数据集 $x=\{(x_1,y_1),(x_2,y_2)...,(x_n,y_n)\}$，为得到一个数据模型可以尽可能准确地预测真实值，构建出线性回归模型：

$$y=\omega^T x+b$$

问题：该模型仅拟合出直线，平面或超平面，不能对数据集拟合出更好的二分类曲线，对此，我们线性回归模型变形，得到

$$y=g^{-1}(\omega^T x+b)$$

这样得到的模型即为“广义线性模型”’，其中 $g(\bullet)$ 为“联系函数”（link function）。

1.2 逻辑回归模型：

逻辑回归模型也即对数回归模型，是广义回归模型在 $g(\bullet)=\ln(\bullet)$ 时的特例。

但线性回归的预测值 $y=\omega^T x+b$ 是实值，需要一个边界决策函数来将实值转化为二分类的 0/1 值。使用理想的阶跃函数：

$$y=\begin{cases}0, z<0;\\ 0.5, z=0;\\ 1, z>0\end{cases}$$

即通过预测出的 z 值的符号判定类别。

问题：单位阶跃函数不连续，且不存在反函数。

1.2.1 解决办法：

使用对数几率函数：

$$y=\frac{1}{1+e^{-z}}$$

替代阶跃函数,该函数的优点是单调可微，任意阶可导，将对数几率函数代入到广义线性模型中得到：

$$y=\frac{1}{1+e^{(-\omega^T x+b)}}$$

将 y 视为样本 xx 作为正例的可能性，$1-y$ 则是 x 作为反例的可能性，对两者比值即为“几率”，对几率取对数得到：

$$\ln\frac{y}{1-y}=\omega^T x+b$$

即为对数几率，即是用线性回归模型的预测结果取逼近真实标记的对数几率，对应的模型即为“对数几率回归”，即逻辑回归模型。构建出模型后利用极大似然法计算出模型参数 ω 和 b 。

1.2.2 极大似然法求 ω 和 b ：

令 $\hat{\omega}=(\omega;b)$ ， $\hat{x}=(x;1)$ ，则 $\omega^T x+b$ 可简写为 $\hat{\omega}^T\hat{x}$ ，

样本为正例的可能性：

$$p(y=1|x)=\frac{e^{\omega^T x+b}}{1+e^{\omega^T x+b}}$$

样本为反例的可能性：

$$p(y=0\mid x)=\frac{1}{1+e^{\omega^T x+b}}$$

为使每个样本属于其真实标记的概率越大越好，求对数似然函数的极大值：

$$(\omega^*,b^*)=\arg\max_{(\omega,b)}\sum_{i=1}^{m}\ln\ p(y_i\mid x_i;\omega,b)$$

$$=\arg\max_{(\omega,b)}\sum_{i=1}^{m}[y_i p_1(\hat{x}_i;\hat{\omega})+(1-y_i)p_0(\hat{x}_i;\hat{\omega})$$

得到高阶可导连续凸函数：

$$\hat{\omega}^*=\arg\min_{\hat{\omega}}\sum_{i=1}^{m}(-y_i\hat{\omega}^T\hat{x}_i+\ln(1+e^{\hat{x}}))$$

最后通过梯度下降法得到 $\hat{\omega}^*$ 。

2 模型训练使用的算法:

2.1 梯度下降法:

梯度下降法是一种常用的一阶优化方法，是求解无约束问题最简单、最经典的方法之一。

2.1.1 随机梯度下降法（SGD）：

在求解对数几率模型问题中，通过随机梯度下降法（SGD）求得 $\hat{\omega}^*$ 的最优解。

输入：目标函数： $E(\hat{\omega})$　　梯度函数： $\nabla E(\hat{\omega}^{(k)})$　计算精度： ε

输出： $E(\hat{\omega})$ 极小值 $\hat{\omega}^*$

算法步骤：

（1） 取初始值 $\hat{\omega}^{(0)}$，k=0；

（2） 计算 $E(\hat{\omega}^{(k)})$；

（3） 计算 $\nabla E(\hat{\omega}^{(k)})$，若 $\|\nabla E(\hat{\omega}^{(k)})\|<\varepsilon$ 时，令 $\hat{\omega}^*=\hat{\omega}^{(k)}$，停止迭代；否则，求 η_k 使 $\min_{\eta\geq 0}E(\hat{\omega}^{(k)}+\eta*(-\nabla E(\hat{\omega}^{(k)})))$；

（4） 置 $\hat{\omega}^{(k+1)} == \hat{\omega}^{(k)} + \eta_k * (-\nabla E(\hat{\omega}^{(k)}))$ ， 计 算 $E(\hat{\omega}^{(k+1)})$ ， 当 $\| E(\hat{\omega}^{(k+1)}) - E(\hat{\omega}^{(k)}) \| < \varepsilon$ 或 $\| \hat{\omega}^{(k+1)} - \hat{\omega}^{(k)} \| < \varepsilon$ 时，令 $\hat{\omega}^{*} = \hat{\omega}^{(k)}$，停止迭代；否则 k=k+1，返回步骤 3。

2.2 交叉熵推导：（以下为 LossFunction 交叉熵的推导）

正常来说，Loss Function 可使用均方误差 $L(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$，但对于逻辑回归模型，则不适合使用，其原因是逻辑回归的 Loss Function 一般不是凸函数，而在使用梯度下降算法时可能得到局部最优解而非全局最优解。故我们需要换一种损失函数去求解。

假设现在的样本为：$D = \{(x_1^T, y_1), (x_2^T, y_2), (x_3^T, y_3) \ldots (x_m^T, y_m)\}, y_i \in \{0,1\}$

由上可知，假设函数可以定义为：$\hat{y}_i = h(x_i) = \frac{1}{1 + e^{-\omega^T x_i + b}}$，由于 y 只能取{0,1}，故对某一个 x_i， y 取值的概率可写为：

$$P(y_i = 1 | x_i) = h(x_i)$$

$$P(y_i = 0 | x_i) = 1 - h(x_i)$$

对概率取对数，可得到：

$$\log P(y_i = 1 | x_i) = \log h(x_i)$$

$$\log P(y_i = 0 | x_i) = \log[1 - h(x_i)]$$

故正确能表征组合正确的对数概率为

$$y_i \log h(x_i) + (1 - y_i) \log[1 - h(x_i)]$$

对总体求和即可得到总体样本的正对表征对数概率，也即代表了整体训练样本的表现能力：

$$\sum_{i=1}^{m} y_i \log h(x_i) + (1 - y_i) \log[1 - h(x_i)]$$

对于损失值，显然表现能力越好，上式的值越大，故用上式的相反数代表损失，再求平均得到平均损失，即得到了交叉熵损失函数：

$$L(y,\hat{y}) = -\frac{1}{m}\sum_{i=1}^{m} y_i \log h(x_i) + (1-y_i)\log[1-h(x_i)], 其中 \hat{y} = h(x)$$

3 绘制 ROC 曲线和 PR 曲线

3.1 PR 曲线

对于二分类问题，可将样例根据其真实类别与学习器预测类别的组合划分为真正例(TP)，真反例(TN)，假正例(FP)，假反例(FN)四种情形。分类结果的混淆矩阵可表示为:

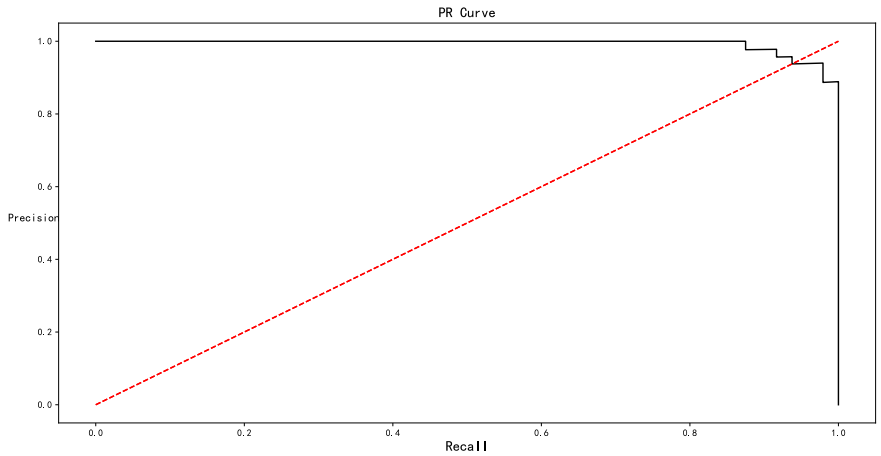
真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

查准率 P(*Precision*)与查全率 R(*Recall*)分别定义为

$$P = \frac{TP}{TP+FP}$$

$$R = \frac{TP}{TP+FN}$$

根据学习器的预测结果按正例可能性大小对样例进行排序,并逐个把样本作为正例进行预测，即可得到查准率——查全率曲线，称为“P-R”曲线，如图(其中红线为平衡线)：



结论：我们可以看出 P-R 曲线整体占的右上角，而且查准率和查全率都在 80%以上，说明学习器性能好，分类的效果较好。

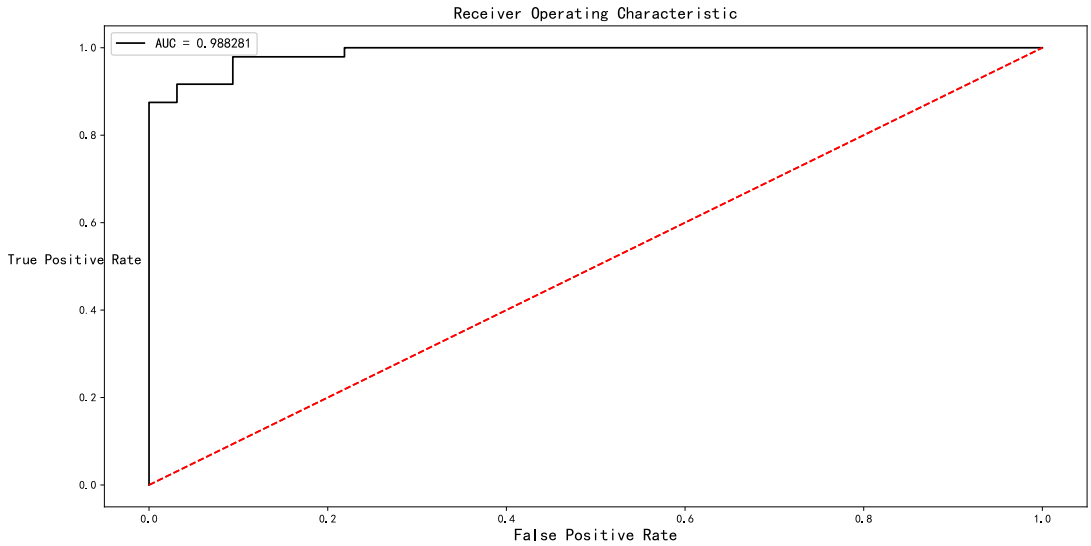
3.2 ROC 曲线

除了 P-R 曲线，ROC 曲线(受试者工作特征)是可以研究学习器泛化性能的重要曲线，其中，ROC 的纵轴为真正例率(TPR)，横轴为假正例率(FPR)，其中，TPR 和 FPR 的定义分别为：

$$TPR = \frac{TP}{TP + FN} = R$$

$$FPR = \frac{FP}{TN + FP}$$

绘制出的 ROC 曲线如图：



AUC 计算：

对于我们绘制的 ROC 曲线，其覆盖的面积值 AUC（Area Under ROC Curve）可以表现出学习器的优劣程度，计算公式为：

$$AUC = \frac{1}{2}\sum_{i=1}^{m-1}(x_{i+1} - x_i)\cdot(y_i + y_{i+1})$$

此处的 AUC 计算是用 1 去减排序损失值，即：

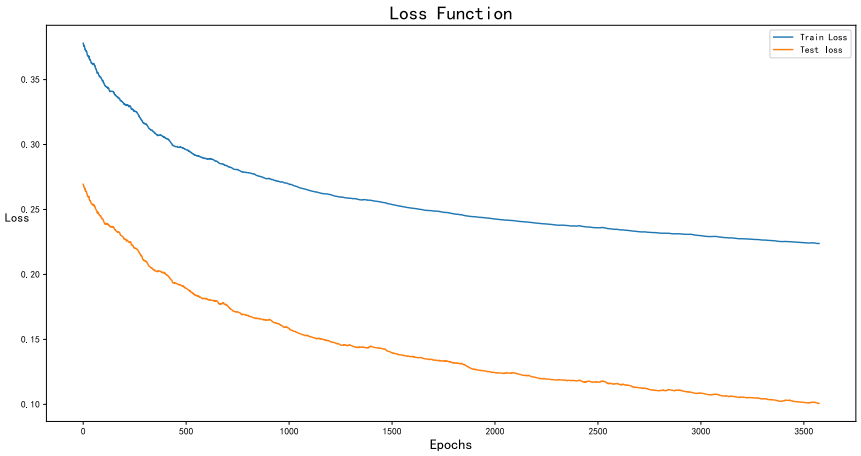
$$AUC = 1 - l_{rank}$$

其中 l_{rank} 代表 ROC 曲线之上的面积，最终计算出：

$$\text{AUC}=0.98828125$$

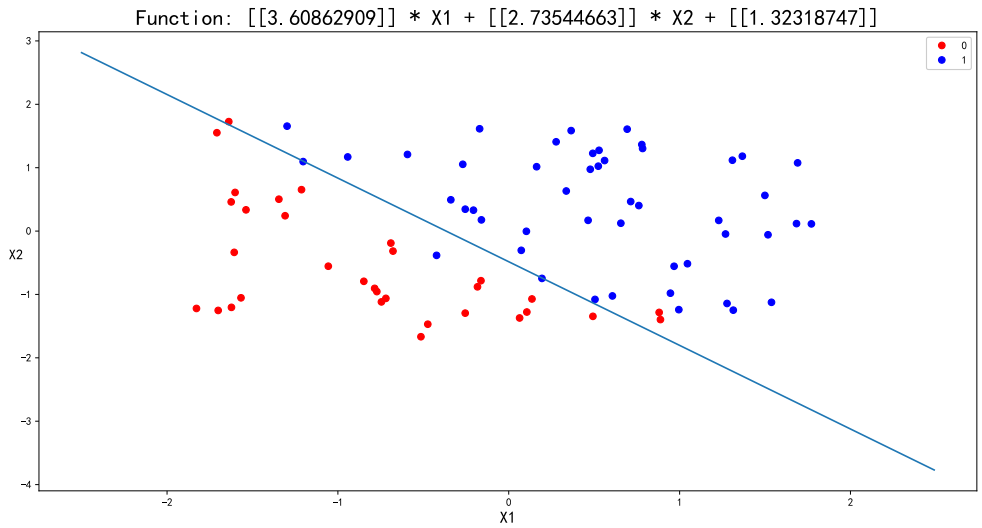
3.3 loss 曲线

根据 2.2 对交叉熵的推导，我们得到该逻辑回归模型的损失函数，绘制出如下图像：



3.4 回归曲线

绘制出的得到的回归曲线如下：



4 模型训练收获：

在机器学习过程中，面对数据处理的问题，通常会遇到类似的数据分类问题，对此我们需要通过现有的数据用某种方法来拟合出一种模型，以达到分类更多数据的目的。

在二分类问题中，为了使线性回归问题中得到的结果是 0/1 的二分形式，我们使用逻辑回归模型对数据建模处理，在建模过程中，为得到更精确的二分模型，需要对模型的构建进行优化，优化过程使用最速下降算法。在最速下降法的优化过程中，我们使用了随机选取样本值的方法，这样能够增加学习器构建的过程中样本的广泛度，以提高学习器的准确性。

在使用最速下降算法对数据处理的过程中，我们需要一组数据来对学习器进行训练，让学习器能预测得到最接近真实值的结果。而得到新的学习器之后，又需要一组新数据来测试学习器是否符合预期。对此问题我们采用对原数据进行划分的操作，即将原数据划分为一定的比例，按照该比例确定每一次的测试集和训练集。而我们将此比例确定为 0.3，即训练集和测试集的比例按照 3:7 来划分，最速下降法的迭代次数设置为 50000，当两次迭代的差值小于某一阈值时，即学习器接近真实值的达到某一程度时，就得到最终的函数，而我们将这一阈值设置在10^{-6}，这样经过多次学习器训练迭代即可得到最接近真实值的学习器。

对于损失函数 Loss Function 的计算，正常来说，Loss Function 可使用均方误差 $L(y,\hat{y})=\frac{1}{2}(y-\hat{y})^2$，但对于逻辑回归模型，则不适合使用，其原因是逻辑回归的 Loss Function 一般不是凸函数，而在使用梯度下降算法时可能得到局部最优解而非全局最优解。故我们需要换一种损失函数去求解。对于这一问题，我们使用交叉熵函数来求得全局最优解。

5 代码:

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

mpl.rcParams["font.sans-serif"] = [u"SimHei"]
mpl.rcParams["axes.unicode_minus"] = False

path = u"第一次作业数据集/3_train.csv"
df = pd.read_csv(path,header=None)
X = df[[0,1]]
Y = df[[2]]
X = np.array(X)
Y = np.array(Y)
mean_X1 = X[:,0].mean()
mean_X2 = X[:,1].mean()
std_X1 = X[:,0].std()
std_X2 = X[:,1].std()
X[:,0] = (X[:,0] - mean_X1) / std_X1
X[:,1] = (X[:,1] - mean_X2) / std_X2
X = np.hstack([X,np.ones((X.shape[0], 1))])
X_train,X_test,Y_train,Y_test = train_test_split(X, Y, test_size=0.1, random_state=0)
loss_list = []
test_loss_list = []
epochs_list = []

def sigmoid(x):
```

```
    """
    return sigmid(x)
    """
    return 1.0 / (1.0 + np.exp(-x))


def loss_func(xMat, yMat, weights):
    """
    计算损失函数
    xMat: 特征数据矩阵
    weights: 参数
    yMat: 标签数据矩阵
    return: 损失函数
    """
    m, n = xMat.shape
    hypothesis = sigmoid(np.dot(xMat, weights))  # 预测值
    loss = (-1.0 / m) * np.sum(yMat.T * np.log(hypothesis) + (1 - yMat).T * np.log(1 - hypothesis))  # 损失函数
    return loss


def SGD(data_x, data_y, test_x, test_y, alpha=0.01, max_epochs=50000,epsilion=1e-8):
    xMat = np.mat(data_x)
    yMat = np.mat(data_y)
    xMat_test = np.mat(test_x)
    yMat_test = np.mat(test_y)
    m, n = xMat.shape
    weights = np.ones((n, 1))  # 模型参数
    epochs_count = 0
    global loss_list
    global test_loss_list
```

```
    global epochs_list
    while epochs_count < max_epochs:
        rand_i = np.random.randint(m)  # 随机取一个样本

        loss = loss_func(xMat, yMat, weights) #前一次迭代的损失值
        test_loss = loss_func(xMat_test, yMat_test, weights)
        hypothesis = sigmoid(np.dot(xMat[rand_i,:], weights)) #预测值
        error = hypothesis - yMat[rand_i,:] #预测值与实际值误差

        grad = np.dot(xMat[rand_i,:].T, error) #损失函数的梯度
        weights = weights - alpha * grad #参数更新
        loss_new = loss_func(xMat, yMat, weights)#当前迭代的损失值
        test_loss_new = loss_func(xMat_test, yMat_test, weights)
        print(loss_new)

        if abs(loss_new-loss)<epsilion:
            break
        loss_list.append(loss_new)
        test_loss_list.append(test_loss_new)
        epochs_list.append(epochs_count)
        epochs_count += 1

    print(f'Final Train Loss = {loss_new}')
    print(f'Final Test Loss = {test_loss_new}')
    print('迭代到第{}次，结束迭代！'.format(epochs_count))
    return weights

def Acc(weights, test_x, test_y):
    xMat_test = np.mat(test_x)
```

```
    m, n = xMat_test.shape
    result = []
    for i in range(m):
        proba = sigmoid(np.dot(xMat_test[i,:], weights)) # 预测值
        if proba < 0.5:
            predict = 0
        else:
            predict = 1
        result.append(predict)
    acc = (np.array(result)==test_y.squeeze()).mean()
    return acc


def GetCurveParams(data_y, y_pred, theta=0.001):
    k = 1 / theta
    m, _ = data_y.shape
    x_ = []
    y_ = []
    z_ = []
    point5_info = {}
    auc = 0.0
    prev_z = 0
    for i in range(1, int(k) + 1):
        tp = 0
        fp = 0
        tn = 0
        fn = 0
        p = theta * i
        for j in range(m):
            if y_pred[0, j] >= p and data_y[j, 0] == 1:
```

```
            tp += 1
        elif y_pred[0, j] >= p and data_y[j, 0] == 0:
            fp += 1
        elif y_pred[0, j] < p and data_y[j, 0] == 1:
            fn += 1
        elif y_pred[0, j] < p and data_y[j, 0] == 0:
            tn += 1
        else:
            pass

# for label 1
recall = tp / (tp + fn)
precision = (tp / (tp + fp)) if (tp + fp) != 0 else 1
fpr = fp / (fp + tn)
f1_score = 2 * precision * recall / (precision + recall)
# for label 0
recall0 = tn    / (tn + fp)
precision0 = (tn / (tn + fn)) if (tn + fn) != 0 else 1
f1_score0 = 2 * precision0 * recall0 / (precision0 + recall0)
x_.append(recall)# Recall & TPR [Ascending]
y_.append(precision) # Precision
z_.append(fpr) # FPR [Descending]

if p == 0.50:
        accuracy = (tp + tn) / (tp + tn + fp + fn)
       point5_info={'tp': tp, 'fp': fp, 'tn': tn, 'fn': fn, 'precision': precision, 'recall': recall,
        'f1-score': f1_score, 'pos': tp + fn, 'precision0': precision0, 'recall0': recall0,
        'f1-score0': f1_score0, 'neg': tn + fp, 'accuracy': accuracy}
```

```
            if z_[-1] != prev_z:
                auc += (z_[-1] - prev_z) * x_[-1] # Inverse Sum-ing
                prev_z = z_[-1]
    point5_info['auc'] = 1 - auc
    return (x_[::-1], y_[::-1], z_[::-1], point5_info)


def Visualizer(data_x, data_y, weights):

    ax = plt.subplot(2, 2, 1)
    ax.plot(data_x[data_y.reshape(-1)==0,0], data_x[data_y.reshape(-1)==0,1], 'or', label='1')
    ax.plot(data_x[data_y.reshape(-1)==1,0], data_x[data_y.reshape(-1)==1,1], 'ob', label='0')
    x = np.arange(-2.5, 2.5, 0.01)
    y = (-weights[2] - weights[0] * x) / weights[1]
    ax.plot(x, y.T)
    plt.legend(loc='best')
    plt.title(f'Function: {weights[0]} * X1 + {weights[1]} * X2 + {weights[2]}')
    plt.xlabel('X1')
    plt.ylabel('X2')

    ax = plt.subplot(2, 2, 2)
    plt.plot(epochs_list, loss_list, label='Train Loss')
    plt.plot(epochs_list, test_loss_list, label='Test loss')
    plt.legend(loc='best')
    plt.title('Loss Function')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
```

```
y_pred = weights[0] * data_x[:,0] + weights[1] * data_x[:,1] + weights[2]
y_pred_log = np.array(sigmoid(y_pred))
Recall, Precision, fpr, point5_info = GetCurveParams(data_y, y_pred_log)
tpr = Recall.copy()
thr = np.sort(y_pred_log)[::-1].tolist()
Recall.append(1)
Precision.append(0)

ax = plt.subplot(2, 2, 3)
plt.plot(Recall, Precision, 'k')
plt.title('PR Curve')
# plt.plot([(0, 0), (1, 1)], 'r--')
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.ylabel('Precision')
plt.xlabel('Recall')

ax = plt.subplot(2, 2, 4)
plt.plot(fpr, tpr, 'k', label='AUC = %.6f'%point5_info['auc'])
plt.title('Receiver Operating Characteristic')
plt.plot([(0, 0),(1, 1)],'r--')
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.legend(loc='best')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')

df = pd.DataFrame(np.array([[point5_info['precision'], point5_info['recall'],
```

```
point5_info['f1-score'], point5_info['pos']],
                                    [point5_info['precision0'],    point5_info['recall0'],
point5_info['f1-score0'], point5_info['neg']]]),
                                    columns=['precision', 'recall', 'f1-score', 'support'],
                                    index=['1.0', '0.0'])
    print(df)
    print('AUC: ', point5_info['auc'])
    print('Accuracy: ', point5_info['accuracy'])
    print('Loss: ', loss_func(data_x, data_y, weights))


    plt.show()


def main():
    weights = SGD(X_train, Y_train, X_test, Y_test)
    acc = Acc(weights, X_test, Y_test)
    print("Test accuracy: ", acc)
    Visualizer(X, Y, weights)


if __name__=='__main__':
    main()
```