第三章 模型评估作业

1. 结合对性能度量部分的阅读，简述错误率、精度、查准率与查全率的含义。

① 错误率与精度

错误率(Error Rate)：是分类错误的样本数占样本总数的比例。

对样例集 D，分类错误率计算公式如下所示。

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(f(\boldsymbol{x}_i)\neq y_i) \tag{1.1}$$

对公式（1.1）解释：统计分类器预测出来的结果与真实结果不相同的个数，然后除以总的样例集 D 的个数。

精度(Accuracy)：是分类正确的样本数占样本总数的比例。

对样例集 D，精度计算公式如下所示。

注意：这里的分类正确的样本数指的不仅是正例分类正确的个数还有反例分类正确的个数。

$$\begin{aligned}\mathrm{acc}(f;D) &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(f(\boldsymbol{x}_i)=y_i)\\ &= 1-E(f;D)\,.\end{aligned} \tag{1.2}$$

对公式（2）的解释：先统计分类正确的样本数，然后除以总的样例集 D 的个数。

② 查准率和查全率

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(true positive)、假正例(false positive)、真反例(true negative)、假反例(false negative)四种情形，令 TP、FP、TN、FN 分别表示其对应的样例数，则显然有 TP+FP+TN+FN=样例总数。分类结果的“混淆矩阵”(confusion matrix)如表 1 所示。

表 1：分类结果混淆矩阵

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(Precision)，又叫准确率，缩写表示用 P。查准率是针对我们预测结果而言的，它表示的是预测为正的样例中有多少是真正的正样例。定义公式如下所示。

$$P=\frac{TP}{TP+FP} \tag{1.3}$$

查全率(Recall)，又叫召回率，缩写表示用 R。查全率是针对我们原来的样本而言的，它表示的是样本中的正例有多少被预测正确。定义公式如下所示。

$$R=\frac{TP}{TP+FN} \tag{1.4}$$

③ F1 度量（题干中没有提到，补充一个度量）

$$F1=\frac{2\times P\times R}{P+R}=\frac{2\times TP}{样本总数+TP-TN} \tag{1.5}$$

2. 数据集包含 1000 个样本，其中 500 个正例、500 个反例，将其划分为包含 70%样本的训练集和 30%样本的测试集用于留出法评估，试估算共有多少种划分方式？

分层抽样：从 500 个正例中，抽 70%做训练集，30%做测试集$500*30\%=150$，共C_{500}^{150}
从 500 个反例中，抽 70%做训练集，30%做测试集$500*30\%=150$，共C_{500}^{150}
排列组合计算可得共$C_{500}^{150}\times C_{500}^{150}$ 种划分方式

3. 已知分类结果混淆矩阵如下，试计算错误率,精度，查准率，查全率，F1。

真实情况	预测结果	
	正例	反例
正例	100	300
反例	200	400

① **错误率：** $E(f;D)=\frac{300+200}{100+300+200+400}=0.5$ (3.1)

② **精度：** $acc(f;D)=1-E(f;D)=1-0.5=0.5$ (3.2)

③ **查准率：** $P=\frac{TP}{TP+FP}=\frac{100}{100+200}=0.333333$ (3.3)

④ **查全率：** $R=\frac{TP}{TP+FN}=\frac{100}{100+300}=0.25$ (3.4)

⑤ **F1：** $F1=\frac{2\times TP}{样本总数+TP-TN}=\frac{2\times 100}{1000+100-400}=0.286$ (3.5)

4. 如下所示 10 个测试样本，’Class’一栏表示每个测试样本的真正标签（P 表示正例；N 表示反例），’Score’表示在某分类器中每个测试样本被预测为正样本的概率：

序号	1	2	3	4	5	6	7	8	9	10
Class	P	N	N	P	P	N	N	P	P	N
Score	0.93	0.85	0.80	0.7	0.55	0.50	0.40	0.3	0.2	0.1

画出 ROC 曲线并计算 AUC 的值。

真正例率（*True Positive Rate*, *TPR*）计算公式：

$$TPR=\frac{TP}{TP+FN} \tag{4.1}$$

假正例率（*False Positive Rate*, *FPR*）计算公式：

$$FPR=\frac{FP}{FP+TN} \tag{4.2}$$

① ROC 曲线的绘制：

- 根据顺序逐个把上述样本作为正例进行预测：
  将每一个正例概率预测值设为分类阈值，若大于阈值则分为正类，否则为反类

每次使用公式（4.1）（4.2）计算出真正例率（*True Positive Rate*, *TPR*）与假正例率（*False Positive Rate*, *FPR*）得到 ROC 曲线上的一个坐标

- 将 FPR 作为横坐标，TPR 作为纵坐标，连接每一个坐标点绘制成曲线图

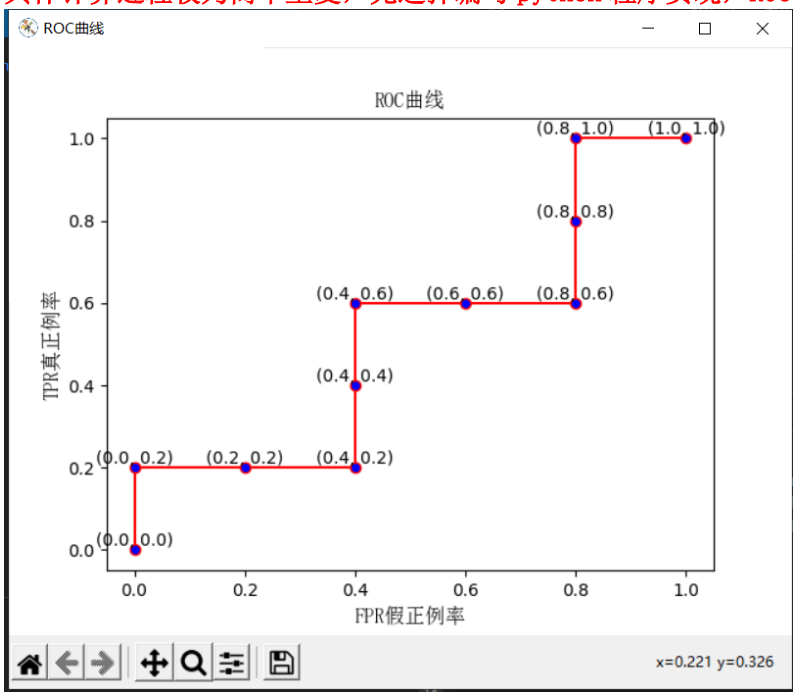
② AUC 值的计算：

从定义可得出，AUC 可以通过对 ROC 曲线下各部分的面积求和可得。假设 ROC 曲线是由坐标为$\{(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)\}$的点按顺序连接形成的图形，则 AUC 可以估算为下列公式：

$$AUC=\frac{1}{2}\sum_{i=1}^{m}(x_{i+1}-x_i)\cdot(y_i+y_{i+1}) \tag{4.3}$$

对 ROC 曲线代入上述公式计算即可得出 AUC 值

具体计算过程较为简单重复，先选择编写 python 程序实现，ROC 曲线结果如下：



AUC 的值为 0.52

```
PS F:\python_code\MachineLearning_task\DecisionTree> python
AUC的值为：  0.52
```

按照 AUC 可以通过对 ROC 曲线下各部分的面积求和验算程序结果步骤如下：，

$AUC=0.4\times0.2+(0.8-0.4)\times0.6+0.2\times1.0=0.52$，AUC 结果正确

附录*python*代码如下：

```
# -*- coding: UTF-8 -*-
import matplotlib.pyplot as plt

Class = ['p', 'n', 'n', 'p', 'p', 'n', 'n', 'p', 'p', 'n']
# 真正标签 P 表示正例；N 表示反例
Score = [0.93, 0.85, 0.80, 0.7, 0.55, 0.50, 0.40, 0.3, 0.2, 0.1]
# 预测为正样本的概率
TP = [0] * 10
FP = [0] * 10
FN = [0] * 10
TN = [0] * 10
TPR = [0.] * 10  # TPR = TP / (TP+FN)
FPR = [0.] * 10  # FPR = FP / (TN+FP)
AUC = 0.0

for i in range(10):
    threshold = Score[i]  # 设置分类阈值为当前score
    for score, clas in zip(Score, Class):
        if (score > threshold):  # score > threshold 预测结果为正例
            if (clas == 'p'):
                TP[i] = TP[i] + 1  # 真实标签为正p，为真正例TP
            else:
                FP[i] = FP[i] + 1  # 真是标签为反f，为假正例FP
        else:  # score <= threshold 预测结果为反例
            if (clas == 'p'):
                FN[i] = FN[i] + 1  # 真实标签为正p，为假反例FN
            else:
                TN[i] = TN[i] + 1  # 真是标签为反f，为真反例TN

for i in range(10):
    TPR[i] = TP[i] / (TP[i] + FN[i])  # TPR = TP / (TP+FN)  y坐标
    FPR[i] = FP[i] / (TN[i] + FP[i])  # FPR = FP / (TN+FP)  x坐标

xy = list(zip(FPR, TPR))
sorted(xy, key=(lambda x: x[0]))
 # xy 数组按照FDR的值进行排序，方便计算AUC
for i in range(9):
    AUC = AUC + 0.5 * (xy[i + 1][0] - xy[i][0]) * (xy[i + 1][1] +
xy[i][1])
    # 计算AUC
print('AUC的值为：', AUC)

plt.figure("ROC曲线")  # 定义一个图像窗口ROC曲线
```

```
plt.plot(FPR, TPR, color='r', markerfacecolor='blue', marker='o')
# 绘制 ROC 曲线
for fpr, tpr in zip(FPR, TPR):  # 绘制每个点的坐标
    plt.text(fpr, tpr, (fpr, tpr), ha='center', va='bottom',
fontsize=10)
plt.xlabel("FPR 假正例率")
plt.ylabel("TPR 真正例率")
plt.show()
print('程序结束')
```