作业 2 分类方法

作者：Zezhong 学号：lightac@mail.ustc.edu.cn

1. 如下表数据，前四列是天气情况（阴晴 outlook，气温 temperature，湿度 humidity，风 windy）；最后一列是类标签，表示根据天气情况是否出去玩。

   (1) “信息熵”是度量样本集合纯度最常用的一种指标，假定当前样本集合 D 中第 k 类样本所占的比例为 p_k（k=1, 2, ···, K），请问当什么条件下，D 的信息熵 Ent(D)取得最大，最大值为多少？
   (2) 根据表中训练数据，基于信息增益决策树应该选哪个属性作为第一个分类属性？
   (3) 对于含有连续型属性的样本数据，决策树和朴素贝叶斯分类能有哪些处理方法？
   (4) 在分类算法的评价指标中，recall 和 precision 分别是什么含义？
   (5) 若一批数据中有 3 个属性特征，2 个类标记，则最多可能有多少种不同的决策树？（不同决策树指同一个样本在两个两个决策下可能得到不同的类标记）

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
rainy	cool	normal	TRUE	no
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
overcast	cool	normal	TRUE	yes
rainy	cool	normal	FALSE	yes
rainy	mild	high	FALSE	yes
overcast	hot	high	FALSE	yes

答：

（1）可知公式$Ent(D) = -\sum_{k=1}^{K} p_k log_2 p_k$。对于这个函数分析可知当$p_k = \frac{1}{k}$时，也就是每类样本都有相同比例时，$Ent(D)$取到最大值，且最大值为$log_2 k$。

（2）分别求各个属性的信息增益。

$$Ent(D) = -(\frac{4}{10} * log_2 \frac{4}{10} + \frac{6}{10} * log_2 \frac{6}{10}) \approx 0.9710$$

$$Ent(D_{outlook}) = -\frac{4}{10} * \left(\frac{3}{4} * log_2 \frac{3}{4} + \frac{1}{4} * log_2 \frac{1}{4}\right) - \frac{4}{10} * \left(\frac{3}{4} * log_2 \frac{3}{4} + \frac{1}{4} * log_2 \frac{1}{4}\right) - \frac{2}{10} * \left(\frac{2}{2} * log_2 \frac{2}{2} + \frac{0}{2} * log_2 \frac{0}{2}\right) \approx 0.6490$$

$$Gain(D, outlook) = Ent(D) - Ent(D_{outlook}) \approx \mathbf{0.3220}$$

$$Ent(D_{temperature}) = -\frac{3}{10} * \left(\frac{2}{3} * log_2 \frac{2}{3} + \frac{1}{3} * log_2 \frac{1}{3}\right) - \frac{4}{10} * \left(\frac{3}{4} * log_2 \frac{3}{4} + \frac{1}{4} * log_2 \frac{1}{4}\right) - \frac{3}{10} * \left(\frac{2}{3} * log_2 \frac{2}{3} + \frac{1}{3} * log_2 \frac{1}{3}\right) \approx 0.8755$$

$$Gain(D, temperature) = Ent(D) - Ent(D_{temperature}) \approx \mathbf{0.0955}$$

$$Ent(D_{humidity}) = -\frac{5}{10} * \left(\frac{3}{5} * log_2 \frac{3}{5} + \frac{2}{5} * log_2 \frac{2}{5}\right) - \frac{5}{10} * \left(\frac{1}{5} * log_2 \frac{1}{5} + \frac{4}{5} * log_2 \frac{4}{5}\right) \approx 0.8464$$

$$Gain(D, humidity) = Ent(D) - Ent(D_{humidity}) \approx \mathbf{0.1246}$$

$$Ent(D_{windy}) = -\frac{7}{10} * \left(\frac{2}{7} * log_2 \frac{2}{7} + \frac{5}{7} * log_2 \frac{5}{7}\right) - \frac{3}{10} * \left(\frac{2}{3} * log_2 \frac{2}{3} + \frac{1}{3} * log_2 \frac{1}{3}\right) \approx 0.8797$$

$Gain(D,windy) = Ent(D) - Ent(D_{windy}) \approx \mathbf{0.0913}$

比较上述计算结果的信息增益可知 outlook 的信息增益最大，因此选择 outlook 属性作为第一个划分属性。

（3）决策树模型而言：对于连续属性，我们可以使用二分法进行离散化。假设连续属性a在样本集D上出现n个不同的取值，从小到大排列，记为$a_1,a_2,\cdots,a_n$。这样可以利用二分法得到元素的候选划分集合$Candidate_a = \left\{\frac{a^i+a^{i+1}}{2} | 1 \le i \le n-1\right\}$。按照这个离散的候选划分集合进行信息增益计算然后再进行划分就可以了。对于朴素贝叶斯分类器：也可以使用将每一个连续的数据离散化，用离散区间代表相应的连续数据。同时，如果我们知道一个数据服从的概率分布，例如高斯分布，我们可以用均值方差进行参数估计，得到特定属性的概率密度。并用它来计算特定属性值的条件概率。

（4）Recall: 预测为正类实际为正类/（预测为正类实际为正类+预测为负类实际为正类）

Precision: 预测为正类实际为正类/（预测为正类实际为正类+预测为正类实际为负类）

（5）从属性的角度考虑：三个属性构成三层决策树，每层有一个属性进行划分因此有$A_3^3 =$ 6种情况。

2. 已知正例点 x1 = (2，3)T，x2 = (3, 2)T，负例点 x3 = (1, 1)T

(1) 试用 SVM 对其进行分类，求最大间隔分离超平面，并指出所有的支持向量。

(2) 现额外有一个点能被 SVM 正确分类且远离决策边界，如果将该点加入到训练集，SVM 的决策边界会受影响吗？为什么？

答：

（1） 构建支持向量机算法，这个过程之中使用 SMO 算法进行优化。通过代码计算可以得到：

最大分离超平面方程为：$0.668y + 0.664x - 2.332 = 0$

支持向量有 3 个：(2 , 3), (3, 2), (1, 1)

最终实现的代码如图所示。

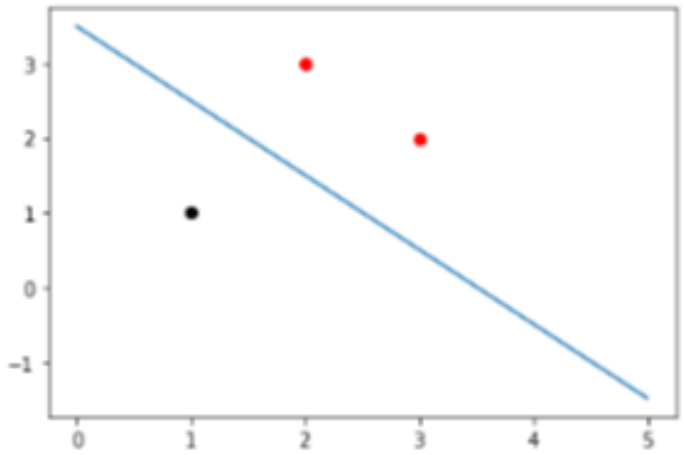
```
class SVM:
    def __init__(self, dataset, label, C, t, iteration):
        self.dataset = dataset
        self.label = label
        self.Numofdataset, self.Dimension = len(self.dataset), len(dataset[0])
        self.C = C
        self.t = t
        self.b = 0
        self.a = np.zeros(self.Numofdataset)
        self.iteration = iteration
        self.w = np.zeros(self.Dimension)
    def Kernel(self, i, j):
        result = np.matmul(self.dataset[i], self.dataset[j].T)
        return result
    def getLH(self, i, j):
        L, H = 0, 0
        if self.label[i] != self.label[j]:
            L = max([0, self.a[j] - self.a[i]])
            H = min([self.C, self.C + self.a[j] - self.a[i]])
        else:
            L = max([0, self.a[j] + self.a[i] - self.C])
            H = min([self.C, self.a[i] + self.a[j]])
        return L, H
```

```
def SMO(self):
    it = 0
    while it < self.iteration:
        isok = 0
        for i in range(self.Numofdataset):
            yi = 0
            for j in range(self.Numofdataset):
                yi += self.a[j] * self.label[j] * np.matmul(self.dataset[i], self.dataset[j].T)
            yi += self.b
            yi -= self.label[i]
            if self.label[i] * yi < -self.t and self.a[i] < self.C or self.label[i] * yi > self.t and self.a[i] > 0:
                while True:
                    j = random.choice(range(self.Numofdataset))
                    if j != i:
                        break
                yj = 0
                for k in range(self.Numofdataset):
                    yj += self.a[k] * self.label[k] * np.matmul(self.dataset[j], self.dataset[k].T)
                yj += self.b
                yj -= self.label[j]
                ai = self.a[i]
                aj = self.a[j]

                L, H = self.getLH(i, j)
                cha = self.Kernel(i, i) + self.Kernel(j, j) - 2 * self.Kernel(i, j)
                if L == H or cha <= 0:
                    continue
                self.a[j] += self.label[j] * (yi - yj) / cha
                if self.a[j] < L:
                    self.a[j] = L
                if self.a[j] > H:
                    self.a[j] = H
                if abs(self.a[j] - aj) < 0.000001:
                    continue
                self.a[i] += self.label[i] * self.label[j] * (aj - self.a[j])
                b1 = self.b - yi - self.label[i] * self.Kernel(i, i) * (self.a[i] - ai) - self.label[j] * self.Kernel(i, j) * (self.a[j] - aj)
                b2 = self.b - yj - self.label[i] * self.Kernel(i, j) * (self.a[i] - ai) - self.label[j] * self.Kernel(j, j) * (self.a[j] - aj)
                if 0 < self.a[i] < self.C:
                    self.b = b1
                elif 0 < self.a[j] < self.C:
                    self.b = b2
                else:
                    self.b = (b1 + b2) / 2.0
                isok = 1
        if isok == 0:
            it += 1
        else:
            it = 0
    for i in range(self.Numofdataset):
        self.w += self.label[i] * self.a[i] * self.dataset[i]
```

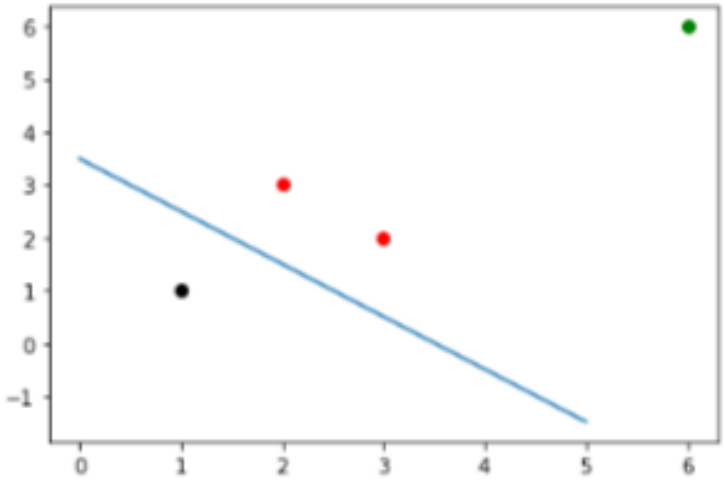
```
if __name__ == '__main__':
    dataset = np.array([[2.0, 3.0], [3.0, 2.0], [1.0, 1.0]]).astype(np.float32)
    label = np.array([[1] ,[1] , [-1]])
    model = SVM(dataset, label, 3, 0.01, 100)
    model.SMO()
    print(model.w)
    print(model.b)
    svm_point = []
    for i in range(3):
        if model.a[i] > 0:
            svm_point.append(i)
    x_point = np.array([i[0] for i in dataset])
    y_point = np.array([i[1] for i in dataset])
    x = np.linspace(0, 5, 50)
    y = -(model.w[0] * x + model.b) / model.w[1]
    plt.scatter(x_point[:2], y_point[:2], color='red')
    plt.scatter(x_point[-1:], y_point[-1:], color='black')
    support_vector = np.array([dataset[i] for i in svm_point])
    print(support_vector)
    plt.plot(x, y)
    plt.show()
```

```
[0.664 0.668]
[-2.332]
[[2. 3.]
 [3. 2.]
 [1. 1.]]
```



（2） 不会产生影响。SVM 优化的目标是优化离超平面最近的点使得这些点的距离尽可能大。由于新加入的点离决策边界远且分类正确，因此加入影响。下面的补充实验图也能说明。其中绿色为原理决策边界且分类正确的点，可以发现学习到的超平面几乎无变化。

```
[0.66496 0.66752]
[-2.33248]
[[2. 3.]
 [3. 2.]
 [1. 1.]]
```



3. 下表是一个由 15 个贷款申请训练数据，数据包括贷款申请人的四个特征属性：分别是年龄，是否有工作，是否有自己的房子以及信贷情况，表的最后一列为类别，是否同意贷款。

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否

1）请根据上表的训练数据，以错误率作为划分标准来构建预测是否进行放贷的决策树。
2）按照所构建的决策树，对属性值为（中年，无工作，无自己的房子，信贷情况好）的申请者是否进行放贷
3）在构建决策树的时候，可能会出现过拟合的问题，有哪些方法可以避免或者解决？
4）对于含有连续型属性的样本数据，决策树有哪些处理方法？

答：

（1）

$$\text{Error}(\text{年龄}) = \frac{1}{3}\times\frac{2}{5}+\frac{1}{3}\times\frac{2}{5}+\frac{1}{3}\times\frac{1}{5}=\frac{1}{3}$$

$$\text{Error}(\text{工作}) = \frac{10}{15}\times\frac{4}{10}+\frac{1}{3}\times 0=\frac{4}{15}$$

$$\text{Error}(\text{有自己房子}) = \frac{3}{5}\times\frac{1}{3}+\frac{2}{5}\times 0=\frac{1}{5}$$

$$\text{Error}(\text{信贷情况}) = \frac{1}{3}\times\frac{1}{5}+\frac{2}{5}\times\frac{1}{3}+\frac{4}{15}\times 0=\frac{1}{5}$$

发现有自己房子和信贷情况两者错误率都最小分成两种情况考虑。

如果选择有自己房子：

$$\text{Error}(\text{年龄}) = \frac{4}{9}\times\frac{1}{4}+\frac{2}{9}\times 0+\frac{3}{9}\times\frac{1}{3}=\frac{2}{9}$$

$$\text{Error}(\text{工作}) = \frac{6}{9}\times 0+\frac{3}{9}\times 0=0$$

$$\text{Error}(\text{信贷情况}) = \frac{4}{9}\times 0+\frac{4}{9}\times\frac{1}{2}+\frac{1}{9}\times 0=\frac{2}{9}$$

工作错误率最小且对应两个特征标签一样，此时决策树终止。得到决策树 A。

如果选择信贷情况：

$$\text{Error}(\text{年龄}) = \frac{5}{11}\times\frac{2}{5}+\frac{3}{11}\times\frac{1}{3}+\frac{3}{11}\times\frac{1}{3}=\frac{4}{11}$$

$$\text{Error}(\text{工作}) = \frac{7}{11}\times\frac{1}{7}+\frac{4}{11}\times 0=\frac{1}{11}$$

$$\text{Error}(\text{房子}) = \frac{8}{11}\times\frac{2}{8}+\frac{3}{11}\times 0=\frac{2}{11}$$

工作错误率最小，选择工作。

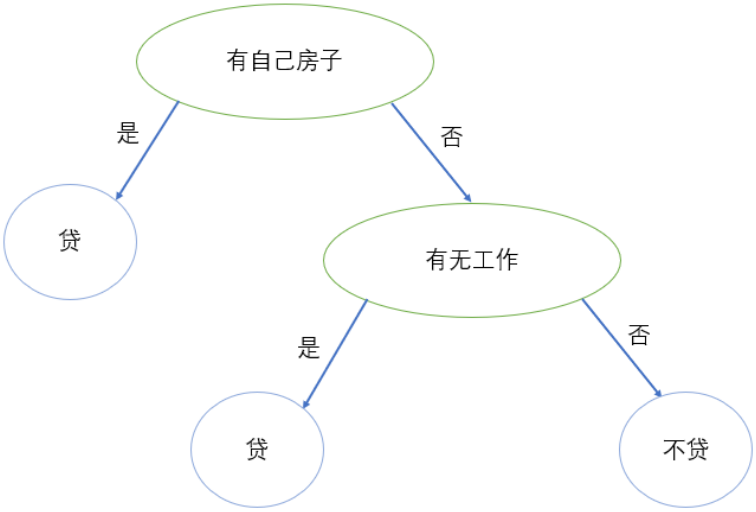
继续划分：

$$\text{Error}(\text{年龄}) = \frac{3}{7}\times 0+\frac{2}{7}\times 0+\frac{2}{7}\times\frac{1}{2}=\frac{1}{7}$$

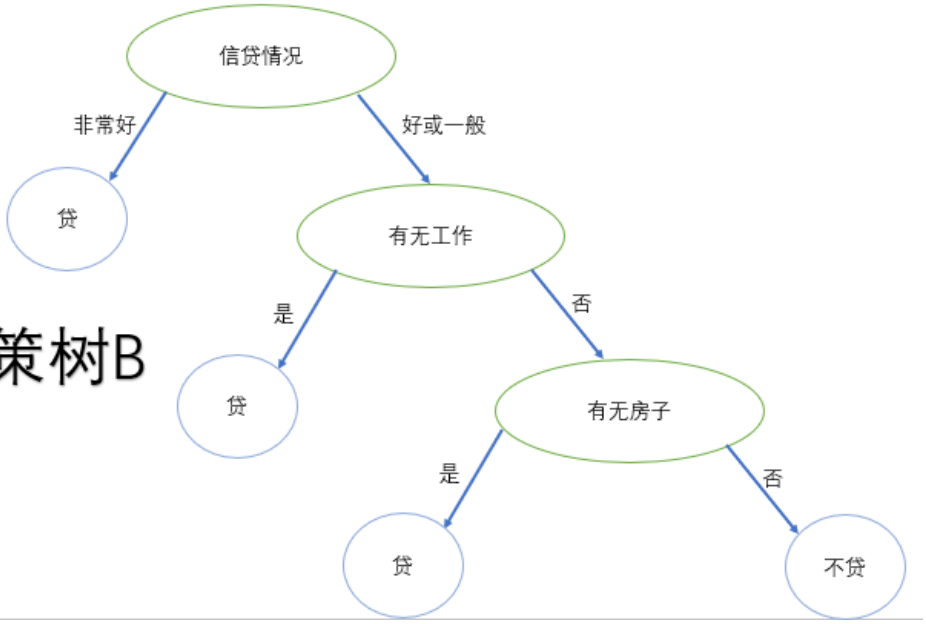
$$\text{Error}(\text{房子}) = \frac{6}{7}\times 0+\frac{1}{7}\times 0=0$$

最后选择房子划分结束。得到决策树 B。

决策树A



决策树B



（2） （中年，无工作，无自己的房子，信贷情况好），由上面的决策树 A 和 B 分析都可知，不贷。

（3） 在构建决策树的过程之中，对于过拟合可以考虑：1.使用后剪枝的方法。2.使用先剪枝的方法，及时停止决策树的生长。3.对于数据集进行更换，使用适合的数据集进行训练。

（4） 该题与第 1 题第 3 问类似。决策树模型而言：对于连续属性，我们可以使用二分法进行离散化。假设连续属性a在样本集D上出现n个不同的取值，从小到大排列，记为$a_1,a_2,\cdots,a_n$。这样可以利用二分法得到元素的候选划分集合$Candidate_a=\left\{\frac{a^i+a^{i+1}}{2}|1\le i\le n-1\right\}$。按照这个离散的候选划分集合进行信息增益计算然后再进行划分就可以了。

4. 请评价两个分类器 M1 和 M2 的性能。所选择的测试集包含 26 个二值属性，记作 A 到 Z。 表中是模型应用到测试集时得到的后验概率（图中只显示正类的后验概率）。因为这是二类问题，所以 P(-)=1-P(+),P(-|A,⋯,Z)=1-P(+|A,⋯,Z)。假设需要从正类中检测实例
   1) 画出 M1 和 M2 的 ROC 曲线（画在一幅图中）。哪个模型更多？给出理由。
   2) 对模型 M1，假设截止阈值 t=0.5。换句话说，任何后验概率大于 t 的测试实例都被看作正例。计算模型在此阈值下的 precision，recall 和 F-score。
   3) 对模型 M2 使用相同的截止阈值重复（2）的分析。比较两个模型的 F-score，哪个模型更好？所得结果与从 ROC 曲线中得到的结论一致吗？
   4) 使用阈值 t=0.1 对模型 M2 重复（2）的分析。t=0.5 和 t=0.1 哪一个阈值更好？该结果和你从 ROC 曲线中得到的一致吗？

实例	真实类	P(+|A,⋯,Z,M1)	P(-|A,⋯,Z,M2)
1	+	0.73	0.61
2	+	0.69	0.03
3	-	0.44	0.68
4	-	0.55	0.31
5	+	0.67	0.45
6	+	0.47	0.09
7	-	0.08	0.38
8	-	0.15	0.05
9	+	0.45	0.01
10	-	0.35	0.04

答：（1）写代码绘制出 M1 和 M2 的 ROC 曲线（没有调库）。首先，预处理对其进行排序。然后循环计算出绘制的点，最后连线，实现的代码如下。

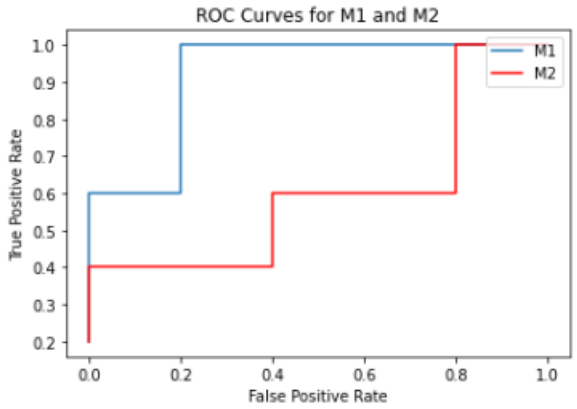
```
import numpy as np
import matplotlib.pyplot as plt
M1_val = np.asarray([0.73, 0.69, 0.67, 0.55, 0.47, 0.45, 0.44, 0.35, 0.15, 0.08])
M1_label = np.asarray([1, 1, 1, 0, 1, 1, 0, 0, 0, 0])

M2_val = np.asarray([0.99, 0.97, 0.96, 0.95, 0.91, 0.69, 0.62, 0.55, 0.39, 0.32])
M2_label = np.asarray([1, 1, 0, 0, 1, 0, 0, 1, 1, 0])

n = len(M1_val)
M1_n_pos = np.sum(M1_label == 1)
M1_n_neg = np.sum(M1_label == 0)
M1_nodex = []
M1_nodey = []
for i in range(n):
    fenzi1 = np.sum(M1_label[0:i+1] == 1)
    fenzi2 = np.sum(M1_label[0:i+1] == 0)
    M1_nodex.append(fenzi2 / M1_n_neg)
    M1_nodey.append(fenzi1 / M1_n_pos)

n = len(M2_val)
M2_n_pos = np.sum(M2_label == 1)
M2_n_neg = np.sum(M2_label == 0)
M2_nodex = []
M2_nodey = []
for i in range(n):
    fenzi1 = np.sum(M2_label[0:i+1] == 1)
    fenzi2 = np.sum(M2_label[0:i+1] == 0)
    M2_nodex.append(fenzi2 / M2_n_neg)
    M2_nodey.append(fenzi1 / M2_n_pos)
plt.title('ROC Curves for M1 and M2')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(M1_nodex, M1_nodey)
plt.plot(M2_nodex, M2_nodey, color = 'red')
plt.legend(['M1','M2'],loc='upper right')
```

绘制出的 ROC 曲线如下图所示。



很容易发现，M1 的 ROC 曲线对于 M2 而言是呈包围的形式，因此围出来的面积（即 AUC）更大，所以 M1 这个模型更强。（因为 AUC 衡量模型好坏的指标，可以判断学习器的优劣）

（2）t = 0.5，

实例	真实类	P(+|A,⋯,Z,M1)	P(-|A,⋯,Z,M2)
1	+	0.73	0.61
2	+	0.69	0.03
3	-	0.44	0.68
4	-	0.55	0.31
5	+	0.67	0.45
6	+	0.47	0.09
7	-	0.08	0.38
8	-	0.15	0.05
9	+	0.45	0.01
10	-	0.35	0.04

对 M1 而言：

Precision = TP/(TP+FP) = 3 / 4

Recall = TP/(TP+FN) = 3 / 5

F1-score = 2 * Precision * Recall / (Precision + Recall) = 2 / 3

（3）

对 M2 而言：

Precision = TP/(TP+FP) = 1 / 2

Recall = TP/(TP+FN) = 4 / 5

F1-score = 2 * Precision * Recall / (Precision + Recall) = 8 / 13

可以发现 M1 的 F1-score 是大于 M2 的，与 ROC 曲线中得到的结论一致。

（4）

t = 0.1 时，

实例	真实类	P(+|A,⋯,Z,M1)	P(-|A,⋯,Z,M2)
1	+	0.73	0.61
2	+	0.69	0.03
3	-	0.44	0.68
4	-	0.55	0.31
5	+	0.67	0.45
6	+	0.47	0.09
7	-	0.08	0.38
8	-	0.15	0.05
9	+	0.45	0.01
10	-	0.35	0.04

对 M1 而言：

Precision = TP/(TP+FP) = 5 / 9

Recall = TP/(TP+FN) = 1

F1-score = 2 * Precision * Recall / (Precision + Recall) = 5 / 7

对 M2 而言：

Precision = TP/(TP+FP) = 1 / 2

Recall = TP/(TP+FN) = 1

F1-score = 2 * Precision * Recall / (Precision + Recall) = 2 / 3

对于 M2 而言 t=0.1 时的 F1-score 会更好。但如果将 t=0.1 的 M2 的 F1-score 与 t=0.5 的 M1 的 F1-score 这两个 F1-score 相等了。也就是可以认为这两个模型差不多好，与 ROC 曲线不一致。

如果将 t=0.1 的 M2 的 F1-score 与 t=0.1 的 M1 的 F1-score 比。发现 M2 的 F1-score 更大。此时与 ROC 曲线不一致，因为会出现召回率都会 1 的情况，这样召回率就没法被有效考虑进去。显然，t=0.5 的阈值会更好。

5. 下图中数据元组已经按分类器返回概率值的递减顺序排序。对于每个元组，计算真正例（TP）、假正例（FP）、真负例（TN）和假负例（FN）的个数。计算真正例率（TPR）和假正例率（FPR）。为该数据绘制 ROC 曲线。

元组号	类	概率
1	P	0.95
2	N	0.85
3	P	0.78
4	P	0.66
5	N	0.60
6	P	0.55
7	N	0.53
8	N	0.52
9	N	0.51
10	P	0.40

答：

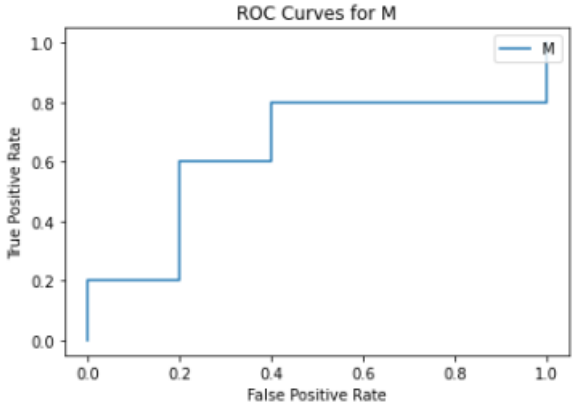
```
import numpy as np
import matplotlib.pyplot as plt
M_val = np.asarray([0.95, 0.85, 0.78, 0.66, 0.60, 0.55, 0.53, 0.52, 0.51, 0.40])
M_label = np.asarray([1, 0, 1, 1, 0, 1, 0, 0, 0, 1])
n = len(M_val)
M_n_pos = np.sum(M_label == 1)
M_n_neg = np.sum(M_label == 0)
TP = []
FP = []
TN = []
FN = []
M_nodex = []
M_nodey = []
for i in range(n):
    fenzi1 = np.sum(M_label[0:i+1] == 1)
    fenzi2 = np.sum(M_label[0:i+1] == 0)
    M_nodex.append(fenzi2 / M_n_neg)
    M_nodey.append(fenzi1 / M_n_pos)
    TP.append(fenzi1)
    FP.append(fenzi2)
    FN.append(M_n_pos - fenzi1)
    TN.append(M_n_neg - fenzi2)
for i in range(n):
  print(str(i) + " " + str(TP[i]) + " " + str(FP[i]) + " " + str(TN[i]) + " " + str(FN[i]) + " " + str(M_nodey[i]) + " " + str(M_nodex[i]))
M_nodex.insert(0, 0)
M_nodey.insert(0, 0)
plt.title('ROC Curves for M')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(M_nodex, M_nodey)
plt.legend(['M'],loc='upper right')
```

通过代码计算出每个元组的真正例（TP）、假正例（FP）、真负例（TN）和假负例（FN）的个数。计算真正例率（TPR）和假正例率（FPR）。如下表所示。

元组号	类	概率	TP	FP	TN	FN	TPR	FPR
1	P	0.95	1	0	5	4	0.2	0
2	N	0.85	1	1	4	4	0.2	0.2
3	P	0.78	2	1	4	3	0.4	0.2
4	P	0.66	3	1	4	2	0.6	0.2
5	N	0.6	3	2	3	2	0.6	0.4
6	P	0.55	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.52	4	4	1	1	0.8	0.8
9	N	0.51	4	5	0	1	0.8	1
10	P	0.4	5	5	0	0	1	1

ROC 曲线如下图所示：

```
0 1 0 5 4 0.2 0.0
1 1 1 4 4 0.2 0.2
2 2 1 4 3 0.4 0.2
3 3 1 4 2 0.6 0.2
4 3 2 3 2 0.6 0.4
5 4 2 3 1 0.8 0.4
6 4 3 2 1 0.8 0.6
7 4 4 1 1 0.8 0.8
8 4 5 0 1 0.8 1.0
9 5 5 0 0 1.0 1.0
<matplotlib.legend.Legend at 0x7f54b4149950>
```



6. 假设两个预测模型 M 和 N 之间进行选择。已经在每个模型上做了 10 轮 10-折交叉验证，其中在第 i 轮，M 和 N 都是用相同的数据划分。M 得到的错误率为 30.5、32.2、20.7、20.6、31.0、41.0、27.7、28.0、21.5、28.0。N 得到的错误率为 22.4、14.5、22.4、 19.6、20.7、20.4、22.1、19.4、18.2、35.0。评述在 1%的显著水平上，一个模型是否显著地比另一个好。

答：

采用 t 检验的策略进行假设检验

原假设 H0：一个模型没有显著地比另一个好

备择假设 H1：一个模型显著地比另一个好

查表可知百分之 1 的显著水平，自由度为 9 的条件下，使用单侧检测。临界值为 3.25。

```python
import numpy as np
M = np.asarray([30.5, 32.2, 20.7, 20.6, 31.0, 41.0, 27.7, 28.0, 21.5, 28.0])
N = np.asarray([22.4, 14.5, 22.4, 19.6, 20.7, 20.4, 22.1, 19.4, 18.2, 35.0])
n = len(M)
Dif = M - N
Dif_mean = np.mean(Dif)
Dif_var = np.var(Dif)
t = abs(0 - Dif_mean) * (n ** 0.5) / (Dif_var ** 0.5)
print(t)
t0 = 3.25
M_mean = np.mean(M)
N_mean = np.mean(N)
if(t < t0):
    print("无显著差别")
elif(M_mean > N_mean):
    print("模型N显著地比M好")
else:
    print("模型M显著地比N好")
```

因为 t = 2.636430214066993 < 3.25。所以不能拒绝原假设，在 1%显著水平上无法判断一个模型是否比另一个模型更好。