一. (30 points) 性能度量

学习器 $\mathcal{L}$ 在某个多分类任务数据集上的预测混淆矩阵如表 1 所示，请回答下列问题。本题的答案请以分式或者小数点后两位的形式给出，比如 P=0.67.

真实情况	**预测结果**		
	第 1 类	**第 2 类**	**第 3 类**
第 1 类	7	1	4
第 2 类	2	6	4
第 3 类	2	2	8

表 1: 学习器 $\mathcal{L}$ 在某个多分类任务数据集上的预测混淆矩阵

1. 该学习器的预测准确率是多少? (5 points)

2. 计算该学习器的微查准率 (micro-P)、宏查准率 (macro-P)、微查全率 (micro-R) 和宏查全率 (macro-R)。(10 points)

3. 计算该学习器的微 F1 (micro-F1) 和宏 F1 (macro-F1)。(5 points)

4. 在多分类中，每个样例只有一个标签。而在多标签分类中，每个样例可以有多个标签。如表 2 所示，一共有 3 个标签，样例 x_1 的标签是 1 和 3，学习器的预测是 1 和 2。请根据教材 2.3.2 查准率、查全率与 F1 章节的描述和表 2 的样例，计算学习器 $\mathcal{L}_1$ 的微查准率、微查全率、微 F1、宏查准率、宏查全率和宏 F1。[提示: 依然是利用各类的混淆矩阵计算微 F1 和宏 F1.] (10 points)

样例	x_1	x_2	x_3	x_4
标签	1,3	1,2	2,3	1,2,3
学习器预测	1,2	2,3	2,3	1,3

表 2: 学习器 $\mathcal{L}_1$ 的样例表

解：

1.

预测正确的样本数为：$7+6+8=21$

总样本数为所有元素的总和：36

因此，预测准确率为：$\text{Accuracy}=\frac{21}{36}\approx 0.58$

该学习器的预测准确率为 $\boxed{0.58}$

2.

微查准率（micro-P）：

$$\text{micro-P}=\frac{\overline{TP}}{\overline{TP}+\overline{FP}}=\frac{7}{12}\approx\boxed{0.58}$$

宏查准率（macro-P）：

$$P_1=\frac{7}{7+2+2}=\frac{7}{11}$$

$$P_2=\frac{6}{6+1+2}=\frac{2}{3}$$

$$P_3=\frac{8}{8+4+4}=\frac{1}{2}$$

$$\text{macro-P}=\frac{1}{3}\sum_{i=1}^{3}P_i=\frac{119}{198}\approx\boxed{0.60}$$

微查全率（micro-R）：

$$\text{micro-R}=\frac{\overline{TP}}{\overline{TP}+\overline{FN}}=\frac{7}{12}\approx\boxed{0.58}$$

宏查全率（macro-R）：

$$R_1=\frac{7}{7+1+4}=\frac{7}{12}$$

$$R_2=\frac{6}{6+4+2}=\frac{1}{2}$$

$$R_3=\frac{8}{8+2+2}=\frac{2}{3}$$

$$\text{macro-R}=\frac{1}{3}\sum_{i=1}^{3}R_i=\frac{7}{12}\approx\boxed{0.58}$$

3.

微 F1（micro-F1）：

$$\text{micro-F1} = \frac{2 \times \text{micro-P} \times \text{micro-R}}{\text{micro-P} + \text{micro-R}} = \frac{7}{12} \approx \boxed{0.58}$$

宏 F1（macro-F1）：

$$\text{macro-F1} = \frac{2 \times \text{macro-P} \times \text{macro-R}}{\text{macro-P} + \text{macro-R}} = \frac{119}{201} \approx \boxed{0.59}$$

4.

以标签 1 为正例，标签 2，3 为反例，对应矩阵如下：

真实情况	**预测结果**	
	正例	**反例**
正例	2（TP）	1（FN）
反例	0（FP）	6（TN）

以标签 2 为正例，标签 1，3 为反例，对应矩阵如下：

真实情况	**预测结果**	
	正例	**反例**
正例	1（TP）	2（FN）
反例	2（FP）	4（TN）

以标签 3 为正例，标签 1，2 为反例，对应矩阵如下：

真实情况	**预测结果**	
	正例	**反例**
正例	1（TP）	2（FN）
反例	2（FP）	4（TN）

由此可得：TP = 4，FP = 4，FN = 5

微查准率（micro-P）：

$$\text{micro-P} = \frac{\overline{TP}}{\overline{TP} + \overline{FP}} = \frac{1}{2} = \boxed{0.50}$$

微查全率（micro-R）：

$$\text{micro-R} = \frac{\overline{TP}}{\overline{TP} + \overline{FN}} = \frac{4}{9} \approx \boxed{0.44}$$

微 F1（micro-F1）：

$$\text{micro-F1} = \frac{2 \times \text{micro-P} \times \text{micro-R}}{\text{micro-P} + \text{micro-R}} = \frac{8}{17} \approx \boxed{0.47}$$

宏查准率（macro-P）：

$$P_1 = 1,\ P_2 = \frac{1}{3},\ P_3 = \frac{1}{3}$$

$$\text{macro-P} = \frac{1}{3}\sum_{i=1}^{3} P_i = \frac{5}{9} \approx \boxed{0.56}$$

宏查全率（macro-R）：

$$R_1 = \frac{2}{3},\ R_2 = \frac{1}{3},\ R_3 = \frac{1}{3}$$

$$\text{macro-R} = \frac{1}{3}\sum_{i=1}^{3} R_i = \frac{4}{9} \approx \boxed{0.44}$$

宏 F1（macro-F1）：

$$\text{macro-F1} = \frac{2 \times \text{macro-P} \times \text{macro-R}}{\text{macro-P} + \text{macro-R}} = \frac{40}{81} \approx \boxed{0.49}$$

二. (30 points) 性能度量

假设数据集包含 10 个样例, 其对应的真实标签和学习器的输出值 (从大到小排列) 如表 3 所示。该任务是一个二分类任务, 标签 1 或 0 表示真实标签为正例或负例。学习器的输出值代表学习器对该样例是正例的置信度 (认为该样例是正例的概率).

样例	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
标签	1	1	0	1	1	0	1	0	0	0
学习器输出值	0.92	0.75	0.62	0.55	0.49	0.4	0.31	0.28	0.2	0.1

表 3: 样例表

1. 计算 P-R 曲线每一个端点的坐标并绘图 (8 points);

2. 计算 ROC 曲线每一个端点的坐标并绘图 (8 points);

3. 基于上一问，计算 AUC 的值 (4 points)；AUC 值请以小数点后两位的形式给出。

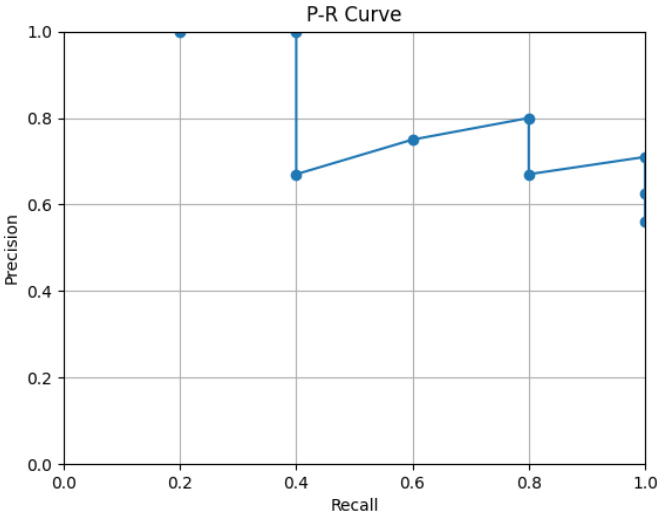
4. FPR95 是一个常见的性能度量指标, 它指的是当真正例率 (true positive rate) 为 95% 时, 假正例率 (false positive rate) 的数值。请问该指标越高学习性能越好还是越低性能越好，并且求解 FPR75 为多少。[提示: FPR75 和 FPR95 类似，FPR75 是真正例率为 75%。](10 points)

解:

1.

阈值	**TP**	**FP**	**FN**	**查准率 P**	**查全率 R**
0.92	0	0	5	Undefined	0
0.75	1	0	4	1	0.2
0.62	2	0	3	1	0.4
0.55	2	1	3	0.67	0.4
0.49	3	1	2	0.75	0.6
0.4	4	1	1	0.8	0.8
0.31	4	2	1	0.67	0.8
0.28	5	2	0	0.71	1
0.2	5	3	0	0.625	1
0.1	5	4	0	0.56	1

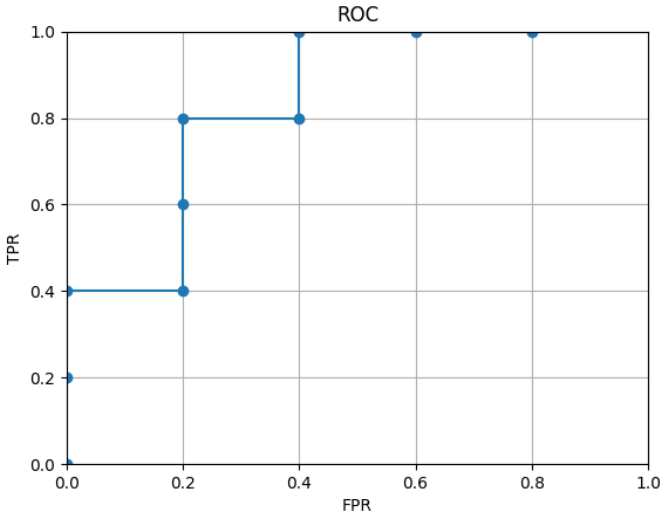
绘制 P-R 曲线如下：



2.

阈值	**TP**	**FP**	**FN**	**TN**	**TPR**	**FPR**
0.92	0	0	5	5	0	0
0.75	1	0	4	5	0.2	0
0.62	2	0	3	5	0.4	0
0.55	2	1	3	4	0.4	0.2
0.49	3	1	2	4	0.6	0.2
0.4	4	1	1	4	0.8	0.2
0.31	4	2	1	3	0.8	0.4
0.28	5	2	0	3	1	0.4
0.2	5	3	0	2	1	0.6
0.1	5	4	0	1	1	0.8

绘制 ROC 曲线如下：



3.

AUC 的值为：

$$\text{AUC} = \frac{1}{2}\sum_{i=1}^{9}(x_{i+1} - x_i) \cdot (y_i + y_{i+1}) = \boxed{0.64}$$

4.

（1）FPR95 越低，表示模型在保持较高 TPR 时能够更好地区分正例和负例，假正例率低，模型性能较好。因此，FPR95 越低越好。

（2）根据 2. 中的结果，TPR $= 0.6$，FPR $= 0.2$；TPR $= 0.8$，FPR $= 0.2$ 因此，FPR75 $= \boxed{0.2}$

三. (10 points) 规范化

Min-max 规范化和 z-score 规范化是两种常见的规范化方法。两种规范化方法分别如下面的公式所示:

$$x' = x'_{min} + \frac{x - x_{min}}{x_{max} - x_{min}} \times (x'_{max} - x'_{min}) \tag{1}$$

$$x' = \frac{x - \bar{x}}{\sigma_x} \tag{2}$$

其中 x 和 x' 分别是规范化前后的值，x_{max} 和 x_{min} 分别是规范化前的最大值和最小值，x'_{max} 和 x'_{min} 分别是规范化后的最大值和最小值，$\bar{x}$ 和 σ_x 是规范化前的均值和标准差。请分析二种规范化的优缺点。

解:

1. Min-max 规范化

优点:

1. 在数据呈线性分布时，Min-max 规范化能较好保持数据的相对大小和比例关系，易于保留原始数据的分布形状
2. 规范化后的数据范围固定在 x'_{min} 到 x'_{max} 之间，容易理解
3. 在一些对输入数据范围有严格要求的算法中（例如神经网络或图像处理），min-max 规范化可以将数据限定在某个区间内，避免过大或过小的输入值影响模型性能。

缺点:

1. Min-max 规范化的计算依赖于数据中的最大值和最小值，若数据中存在异常值，会极大影响规范化结果，导致数据失真
2. 当数据集更新或者扩展时，需要重新计算规范化参数，不利于动态数据场景。

2. z-score 规范化

优点:

1. z-score 规范化基于均值和标准差，异常值对均值和标准差的影响较小
2. z-score 规范化不限制数据的范围，可以处理负值数据，适用范围广

缺点:

1. 需要计算均值和标准差，计算量较大。
2. 对数据的正态分布有一定要求，非正态分布时效果不佳。

四. (30 points) 线性回归

给定包含 m 个样例的数据集 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_m, y_m)\}$，其中 $\mathbf{x}_i = (x_{i1}; x_{i2}; \cdots; x_{id}) \in \mathbb{R}^d$，$y_i \in \mathbb{R}$ 为 $\mathbf{x}_i$ 的实数标记。线性回归模型要求该线性模型的预测结果和其对应的标记之间的误差之和最小:

$$(\mathbf{w}^*, b^*) = \underset{(\mathbf{w},b)}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{m} (y_i - (\mathbf{w}^\top \mathbf{x}_i + b))^2 \tag{3}$$

即寻找一组权重 $(\mathbf{w}, b)$，使其对 D 中示例预测的整体误差最小。定义 $y = [y_1; y_2; \cdots; y_m] \in \mathbb{R}^m$ 且 $\mathbf{X} = [\mathbf{x}_1^\top; \mathbf{x}_2^\top; \cdots; \mathbf{x}_m^\top] \in \mathbb{R}^{m \times d}$.

1. 请将线性回归的优化过程使用矩阵进行表示; (10 points)

2. 使用矩阵形式给出权重 $\mathbf{w}^*$ 和偏移 b^* 最优解的表达式；(10 points)

3. 针对波士顿房价预测数据集 (boston), 编程实现原始线性回归模型。请基于闭式解在训练集上构建模型, 计算测试集上的均方误差 (Mean Square Error, MSE)。请参考如下形式完成函数 linear_regression 和 MSE 的代码。除示例代码中使用到的 sklearn 库函数外, 不能使用其他的 sklearn 函数, 需要基于 numpy 实现线性回归模型闭式解以及 MSE 的计算. (10 points)

In [3]:
```
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split

X, y = load_boston(return_X_y=True)
trainx , testx , trainy , testy = train_test_split(X, y, test_size = 0.33, random_state =42)

def linear_regression(X_train , y_train):
''' 线性回归
    : 参数X_train: np.ndarray , 形状为(n, d), n 个d 维训练样本
    : 参数y_train: np.ndarray , 形状为(n, 1), 每个样本的标签
    : 返回: 权重矩阵W
'''

def MSE(X_train , y_train , X_test , y_test):
''' 调用linear_regression得到权重矩阵W后计算MSE
    : 参数X_train: np.ndarray , 形状为(n, d), n 个d 维训练样本
    : 参数y_train: np.ndarray , 形状为(n, 1), 每个训练样本的标签
    : 参数X_test: np.ndarray , 形状为(m, d), m 个d 维测试样本
    : 参数y_test: np.ndarray , 形状为(m, 1), 每个测试样本的标签
    : 返回: 标量, MSE 值
'''


linear_regression_MSE= MSE(trainx , trainy , testx , testy)
```

解:

1.

将权重向量 w 和偏置 b 合并成一个扩展的权重向量，定义：$\hat{\mathbf{w}} = (\mathbf{w}; b)$

相应地，我们需要扩展矩阵 X，在每个样本后面添加一个 1：$\hat{\mathbf{X}} = [\mathbf{X}, \mathbf{1}] \in \mathbb{R}^{m\times(d+1)}$，其中 $\mathbf{1}$ 是一个 m 维的全 1 列向量。

现在，我们可以将优化目标重写为矩阵形式：

$$\boxed{\hat{\mathbf{w}}^* = \underset{\hat{\mathbf{w}}}{\operatorname{argmin}} \frac{1}{2}(\mathbf{y} - \hat{\mathbf{X}}\hat{\mathbf{w}})^{\mathrm{T}}(\mathbf{y} - \hat{\mathbf{X}}\hat{\mathbf{w}})}$$

2.

由 1. 得

$$\hat{\mathbf{w}}^* = \underset{\hat{\mathbf{w}}}{\operatorname{argmin}} \frac{1}{2}(\mathbf{y} - \hat{\mathbf{X}}\hat{\mathbf{w}})^{\mathrm{T}}(\mathbf{y} - \hat{\mathbf{X}}\hat{\mathbf{w}})$$

其中 $\hat{\mathbf{w}} = [\mathbf{w}; b]$，$\hat{\mathbf{X}} = [\mathbf{X}, \mathbf{1}]$。

令 $E_{\hat{\mathbf{w}}} = (\mathbf{y} - \hat{\mathbf{X}}\hat{\mathbf{w}})^{\mathrm{T}}(\mathbf{y} - \hat{\mathbf{X}}\hat{\mathbf{w}})$，则 $E_{\hat{\mathbf{w}}} = (\mathbf{y}^{\mathrm{T}}\mathbf{y} - \mathbf{y}^{\mathrm{T}}\hat{\mathbf{X}}\hat{\mathbf{w}} - \hat{\mathbf{w}}^{\mathrm{T}}\hat{\mathbf{X}}^{\mathrm{T}}\mathbf{y} + \hat{\mathbf{w}}^{\mathrm{T}}\hat{\mathbf{X}}^{\mathrm{T}}\hat{\mathbf{X}}\hat{\mathbf{w}})$

对 $\hat{\mathbf{w}}$ 求导得：

$$2\hat{\mathbf{X}}^{\mathrm{T}}(\hat{\mathbf{X}}\hat{\mathbf{w}} - \mathbf{y}) = 0$$

解得：

$$\hat{\mathbf{w}} = (\hat{\mathbf{X}}^{\mathrm{T}}\hat{\mathbf{X}})^{-1}\hat{\mathbf{X}}^{\mathrm{T}}\mathbf{y}$$

因为：

$$\hat{\mathbf{w}}^* = [\mathbf{w}^*; b^*] = (\hat{\mathbf{X}}^{\mathrm{T}}\hat{\mathbf{X}})^{-1}\hat{\mathbf{X}}^{\mathrm{T}}\mathbf{y}$$

所以：

$$\begin{bmatrix}\mathbf{w}^* \\ b^*\end{bmatrix} = \begin{bmatrix}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}} & (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{1} \\ \mathbf{1}^{\mathrm{T}}\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}} & \mathbf{1}^{\mathrm{T}}\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{1}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{X}^{\mathrm{T}}\mathbf{y} \\ \mathbf{1}^{\mathrm{T}}\mathbf{y}\end{bmatrix}$$

因此：

$$\boxed{\mathbf{w}^* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}(\mathbf{y} - \mathbf{1}b^*)}$$

$$\boxed{b^* = \frac{1}{m}(\mathbf{1}^{\mathrm{T}}\mathbf{y} - \mathbf{1}^{\mathrm{T}}\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y})}$$

3.

```python
import numpy as np
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split

def linear_regression(X_train, y_train):
    X_train_bias = np.column_stack((X_train, np.ones(X_train.shape[0])))

    W = np.linalg.inv(X_train_bias.T @ X_train_bias) @ X_train_bias.T @ y_train

    return W

def MSE(X_train, y_train, X_test, y_test):
    W = linear_regression(X_train, y_train)

    X_test_bias = np.column_stack((X_test, np.ones(X_test.shape[0])))

    y_pred = X_test_bias @ W

    mse = np.mean((y_test - y_pred)**2)

    return mse

X, y = load_boston(return_X_y=True)
trainx, testx, trainy, testy = train_test_split(X, y, test_size=0.33, random_state=42)

trainy = trainy.reshape(-1, 1)
testy = testy.reshape(-1, 1)

linear_regression_MSE = MSE(trainx, trainy, testx, testy)
print(f"Linear Regression MSE: {linear_regression_MSE}")
```

```
PS E:\大三上\机器学习\作业\作业一> python 1.py
Traceback (most recent call last):
  File "E:\大三上\机器学习\作业\作业一\1.py", line 2, in <module>
    from sklearn.datasets import load_boston
  File "C:\Python312\Lib\site-packages\sklearn\datasets\__init__.py", line 156, in __getattr__
    raise ImportError(msg)
ImportError:
`load_boston` has been removed from scikit-learn since version 1.2.

The Boston housing prices dataset has an ethical problem: as
investigated in [1], the authors of this dataset engineered a
non-invertible variable "B" assuming that racial self-segregation had a
positive impact on house prices [2]. Furthermore the goal of the
research that led to the creation of this dataset was to study the
impact of air quality but it did not give adequate demonstration of the
validity of this assumption.

The scikit-learn maintainers therefore strongly discourage the use of
this dataset unless the purpose of the code is to study and educate
about ethical issues in data science and machine learning.

In this special case, you can fetch the dataset from the original
source::

    import pandas as pd
    import numpy as np

    data_url = "http://lib.stat.cmu.edu/datasets/boston"
    raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
    data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
    target = raw_df.values[1::2, 2]

Alternative datasets include the California housing dataset and the
Ames housing dataset. You can load the datasets as follows::

    from sklearn.datasets import fetch_california_housing
    housing = fetch_california_housing()

for the California housing dataset and::

    from sklearn.datasets import fetch_openml
    housing = fetch_openml(name="house_prices", as_frame=True)

for the Ames housing dataset.

[1] M Carlisle.
"Racist data destruction?"
<https://medium.com/@docintangible/racist-data-destruction-113e3eff54a8>

[2] Harrison Jr, David, and Daniel L. Rubinfeld.
"Hedonic housing prices and the demand for clean air."
Journal of environmental economics and management 5.1 (1978): 81-102.
<https://www.researchgate.net/publication/4974606_Hedonic_housing_prices_and_the_demand_for_clean_air>
```