**模式识别与机器学习 – 实验报告1**

姓名: 顾凯喻 学号: 23307130015 专业：计算机科学与技术 学院：计算机科学与技术学院

本实验聚焦于课程第三讲决策树与最近邻问题的内容，包含以下三部分：

* 分类评测指标 (20%)
* 决策树 (40%)
* 最近邻问题 (40%)

占课程总分：20% | 提交截至日期：**10/13 23:59**

**分类评测指标 – 20%**

1. **指标定义 – 5%**

用你的话简单解释下列每个指标的含义以及使用场景

|  |  |  |
| --- | --- | --- |
|  | 含义 | 使用场景 |
| Accuracy | 正确率，衡量true和predict两者之前的差异 | 分类问题，用于计算最终得到的predict结果和实际true label之间的差距 |
| MSE | 均方误差，平均平方差 | 适用于回归任务 |
| Precision | True\_positive/(true\_positive+false\_positive) | 预测结果符合(positive)的预测正确率 |
| Recall | Tp/(tp+fn) | 一共有多少个positive样本，然后预测时找回了几个 |
| F1 | Precision和recall的调和平均 | 混淆矩阵综合评价 |

1. **代码实现 (见Part1\_ReadMe.md) – 10%**

请在报告中贴出你实现的五个核心函数的代码并简单说明实现的逻辑：

accuracy\_score()

实现逻辑：计算y\_true和y\_pred之间不同的值（错误预测）的数量，得到准确率

accuracy = -1

    y\_true = np.asarray(y\_true).reshape(-1)

    y\_pred = np.asarray(y\_pred).reshape(-1)

    # reshape已经将array展平，shape为(N,)里面每一行都是scalar标量

    if y\_true.shape[0] != y\_pred.shape[0]:

        raise ValueError("y\_true and y\_pred must have the same length.")

    wrong\_times = 0

    # =============== TODO (students) ===============

    for iter in range(y\_true.shape[0]):

        if y\_true[iter] != y\_pred[iter]:

            wrong\_times += 1

    accuracy = 1- wrong\_times / y\_true.shape[0]

    return accuracy

    # ===============================================

mean\_squared\_error()

直接按照MSE的公式展开计算

    error = -1

    y\_true = np.asarray(y\_true, dtype=np.float64).reshape(-1)

    y\_pred = np.asarray(y\_pred, dtype=np.float64).reshape(-1)

    if y\_true.shape[0] != y\_pred.shape[0]:

        raise ValueError("y\_true and y\_pred must have the same length.")

    # =============== TODO (students) ===============

    error = (((y\_true-y\_pred)\*\*2).sum()) / y\_pred.shape[0]

    return error

    # ===============================================

precision\_score()

利用\_binary\_counts()当中broadcast比较得出的tp,fp值代入计算precision

    tp, fp, \_, \_ = \_binary\_counts(y\_true, y\_pred)

    # =============== TODO (students) ===============

    precision = tp/(tp+fp)

    return precision

    # ===============================================

recall\_score()

利用\_binary\_counts()当中broadcast比较得出的tp,fp值代入计算recall

    """Recall = TP / (TP + FN)"""

    tp, \_, fn, \_ = \_binary\_counts(y\_true, y\_pred)

    # =============== TODO (students) ===============

    recall = tp / (tp + fn)

    return recall

f1\_score()

利用前面计算的precision和recall得出f1

    """F1 = 2 \* (P \* R) / (P + R)"""

    # =============== TODO (students) ===============

    precision = precision\_score(y\_true,y\_pred)

    recall = recall\_score(y\_true,y\_pred)

    f\_one = (2\*precision\*recall) / (recall+precision)

    return f\_one

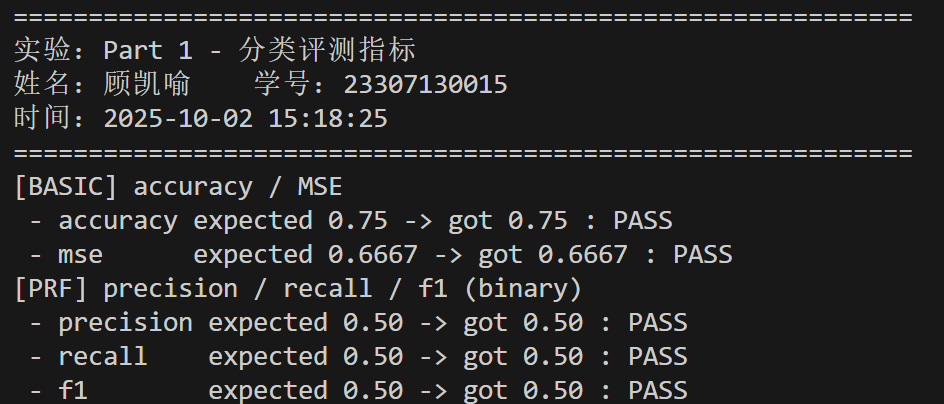
出于篇幅限制，请尽量只选取你完成的最核心的代码进行展示，

可以从vscode中复制，也可以粘贴截图，

在解释时说明涉及到参数的含义，以及你完成代码的逻辑。之后的代码实现部分同理

1. **测试与结果 – 5%**

完成评测指标的函数后，运行 test.py，把输出log的截图粘贴在下面一行：



删掉上面的示例图片，改成你自己的结果，记得在test.py中填上你的名字和学号

使用以下新的输入测试，你可以再test.py中更改，或自己计算

|  |  |
| --- | --- |
| Y\_true | Y\_predict |
| [0, 1, 0, 1, 1, 0, 1, 0, 0] | [1, 1, 1, 1, 1, 0, 0, 0, 0] |

Confusion Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| TP | FP | FN | TN |
| 3 | 2 | 1 | 3 |

指标计算

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Accuracy | MSE | Precision | Recall | F1 |
| 2/3 | 1/3 | 0.6 | 0.75 | 2/3 |

**决策树 – 40%**

1. 简要解释下决策树，以及其优缺点 – 2%

|  |  |
| --- | --- |
| 决策树概述 | 优缺点 |
|  |  |

1. **代码实现（见Part2\_ReadMe.md） – 15%**

请在报告中贴出你实现的四个核心函数的代码并简单说明实现的逻辑：

criterion.py

* \_\_info\_gain(...)
* \_\_info\_gain\_ratio(...)
* \_\_gini\_index(...)
* \_\_error\_rate(...)

1. **测试和可视化 – 15%**

完成criterion.py的四个函数后，运行test\_decision\_tree.py，将会输出对应的accuracy和四张图片，只需要把图片粘贴在下面，每张图的图注写明：**Accuracy、树深度、叶子数**



iris\_error\_rate

Accuracy = | Tree\_depth = | Tree\_leaf\_num =



iris\_gini

Accuracy = | Tree\_depth = | Tree\_leaf\_num =



iris\_info\_gain

Accuracy = | Tree\_depth = | Tree\_leaf\_num =



iris\_info\_gain\_ratio

Accuracy = | Tree\_depth = | Tree\_leaf\_num =

删掉上面的示例图片，改成你自己的结果

1. **进一步探索 – 8%**

本部分希望同学们在固定训练/验证/测试划分下，调参使测试集 Accuracy 尽可能高。

下表给出decision\_tree.py中的可调参数

|  |  |  |
| --- | --- | --- |
| 可调的参数 | 参数说明 | 可调值 |
| criterion | 分裂度量不同，偏好不同；可先粗选再细调 | {info\_gain, info\_gain\_ratio, gini, error\_rate} |
| splitter | random 随机阈值更具多样性，配合多次 seed 取较稳的方案 | {best, random} |
| max\_depth | 控制树深，限制过拟合 | {None, 2-10} |
| min\_samples\_split | 节点最小样本数，越大越保守 | {2, 3, 4, 5, 10, 20…..} |
| min\_impurity\_split | 最小分裂增益阈值，越高越保守 | {0, 1e-4, 1e-3, 1e-2…..} |
| max\_features | 每次候选的特征子集大小，和随机森林思想类似 | { None, "sqrt", "log2", 1..d, 0.5..1.0} |

建议的搜索顺序

1. 先在 {criterion}×{splitter} 上做粗选
2. 固定上一步较优组合，再对 {max\_depth,min\_samples\_split,min\_impurity\_split} 做粗到细的网格
3. 最后微调 max\_features，在不降 Acc 的前提下简化树（更浅/更少叶）

你可以在下表中进行参数的尝试（至少3组），准确率**至少要95%以上**，越高越好

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| criterion |  |  |  |  |
| splitter |  |  |  |  |
| max\_depth |  |  |  |  |
| min\_samples\_split |  |  |  |  |
| min\_impurity\_split |  |  |  |  |
| max\_features |  |  |  |  |
| **accuracy** |  |  |  |  |

高亮你选中的最佳参数组合，在下面粘贴输出的树图



删掉上面的示例图片，改成你自己的结果

**最近邻问题 – 40%**

1. **简要解释下knn算法，以及其优缺点 – 2%**

|  |  |
| --- | --- |
| knn概述 | 优缺点 |
|  |  |

1. **代码实现 (见Part3\_ReadMe.md) – 15%**

请在报告中贴出你实现的三个核心函数的代码并简单说明实现的逻辑：

pairwise\_dist()

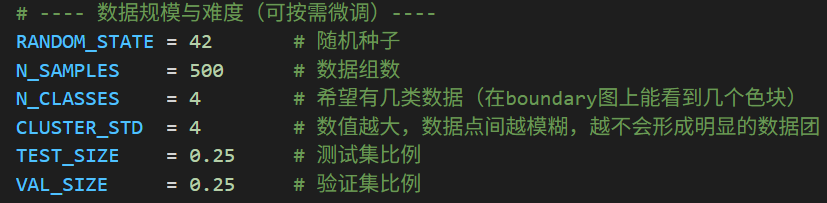
* L2 – twoloop
* L2 – noloop
* Cosine

knn\_predict()

select\_k\_by\_validation()

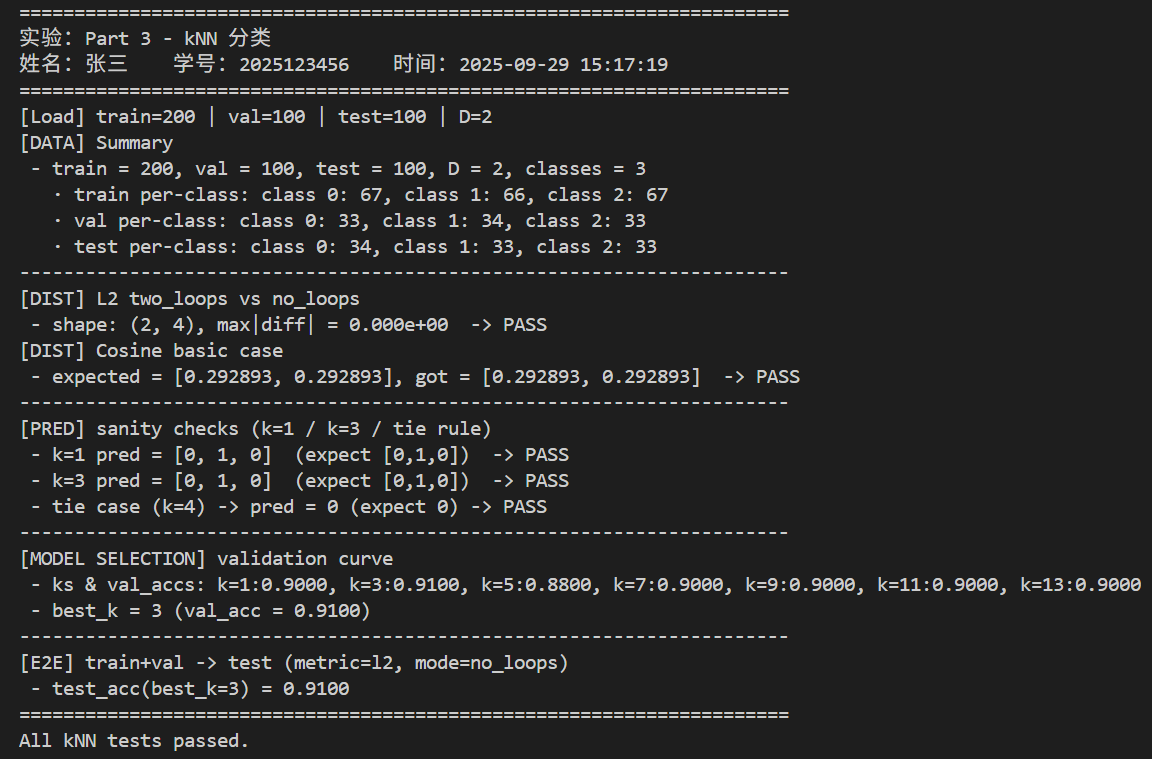
1. **测试与结果可视化 – 15%**

下面是我们在固定测试中使用的数据集参数设定，你可以在data\_generate.py中查看和改变这些参数：



**3.1**

完成knn\_student.py的代码块后，在data\_generate.py中设置以上的参数，然后运行test\_knn.py，把输出的log粘贴在下面，

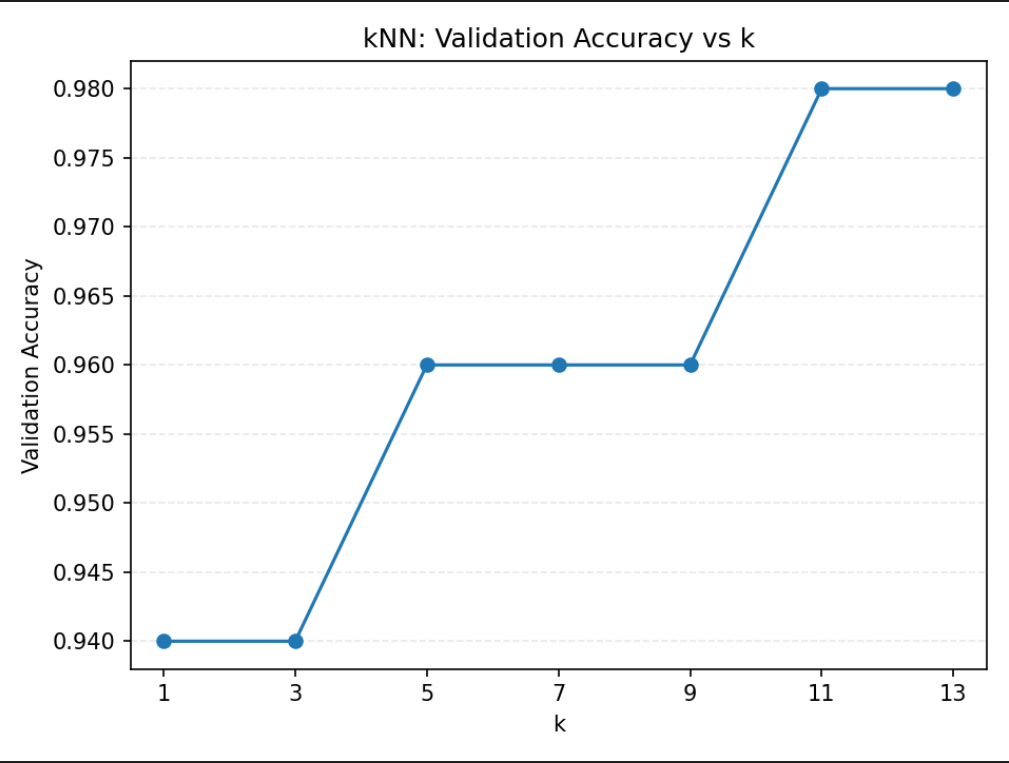


删掉上面的示例图片，改成你自己的结果，记得在test\_knn.py中填上你的名字和学号

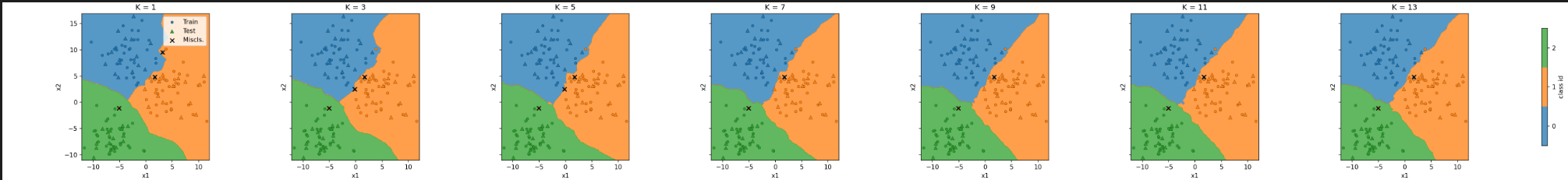
**3.2**

当所有的测试都通过后，运行knn\_student.py，程序会调用viz\_knn.py中的可视化函数，输出knn\_k\_curve.png和knn\_boundary\_grid.png两个图像。需要你在knn\_student.py中修改matric参数，分别生成使用‘L2’和‘cosine’的图像，贴在下面。

1. Matric = ‘L2’



knn\_k\_curve



knn\_boundary\_grid.

1. Metric = ‘cosine’

同L2格式

注意示例图片仅供参考，不是要求的数据集结果，删掉上面的示例图片，改成你自己的结果

**3.3**

观察你得到的测试报告和图像，回答以下的问题：

1. 准确率最高的k值是多少？

|  |  |  |
| --- | --- | --- |
|  | Best\_k | Accuracy |
| L2 |  |  |
| Cosine |  |  |

1. K 对边界的影响，K=1 时边界为何“锯齿/细碎”？K 增大为何更平滑？
2. 在相同的数据下‘L2’和‘cosine’有什么差异（结合图像解释）？他们各自测量的‘距离’是什么？
3. **进一步探索 - 8%**

本部分希望同学们能够选择自己感兴趣的问题进行探索，并完成一份简单的实验报告，我们提供三个样例问题，同学们可以选择其中之一进行探索，更加鼓励自己寻找一个问题进行实验。

样例问题1：探索适合‘L2’和‘Cosine’的数据场景

通过修改data\_generate.py中的数据参数，和k的选择，分别搜索能够在‘L2’和‘Cosine’方法下达到高准确度(95%以上)的数据集，分析两种距离计算方式适配的场景和数据结构。

样例问题2：类内方差（重叠程度）对 k 的影响

通过修改data\_generate.py中的CLUSTER\_STD参数，探索其和k的联系。可以通过下面的问题展开：

CLUSTER\_STD ↑（更模糊）时，best\_k 是否趋向更大？

为什么从“锯齿→平滑”的边界有助于抗噪？

对比 k=1 与 k=best\_k 的误分类点分布（图表 ×），哪些区域最难？

样例问题3：探索数据结构与过拟合/欠拟合的关系

过拟合/欠拟合的定义以及他们呈现的结果是什么样的？

什么样的参数会导致过拟合/欠拟合的发生？

关于简易实验报告，必须包含下面的几个部分：

具体的探索的问题

探索的方法（修改了哪些参数，为什么需要改这些参数）

输出的结果（包括图像和log）

最终的结论

**提交**

* 完成后删除所有红色字体的提示部分，不要改动黑色字体的题干部分
* 选择合适的字体和行间距，保证美观和可读性
* 保证粘贴的图像大小合适，图中内容清晰可见
* 完成后导出为**pdf**，把文件名改为 **PRML-实验1-姓名** ，提交到elearning上，**不需要**提交单独的图像，代码文件或压缩包
* 截至日期在开头