《机器学习》实验一：基于 kNN 的手写数字识别（kNN + LOO）

1. 实验目的

- 实现 k 近邻（kNN）分类器；
- 掌握留一法（LOO）交叉验证；
- 对比自实现与 Weka IBk 的 LOO 精度；
- 学会用图表展示混淆矩阵并形成可复现实验报告。

2. 数据集

- Semeion Handwritten Digit Data Set（UCI）
- 文件： `data/semeion.data.txt` （或将 `semeion_train.txt + semeion_test.txt` 在内存合并）
- 结构：前 256 列为 16×16 展平像素（0/1），后 10 列为独热标签；标签通过 `argmax` 转为 0~9。

3. 环境与依赖

- OS：WSL Ubuntu 24.04
- Python：3.12（Poetry 管理）
- 依赖：numpy、matplotlib（CPU）；Weka CLI（apt 包 weka）
- 日志与图表： `logs/` 、 `reports/figures/` 自动生成

运行示例：

```
# 自实现（k=1,3,5）
poetry run python -m src.exp_1 --data data/semeion.data.txt --k 1 3 5 --seed 2025 --tie min --sa

# Weka CLI（k=1,3,5）
poetry run python -m src.weka_eval --data data/semeion.data.txt --k 1 3 5 --seed 2025 --save-jso
```

4. 方法实现

- 距离：欧氏距离（向量化），在 LOO 中将自样本距离设为 +inf 防止泄漏；
- 投票： `np.bincount` 多数表决，平票策略默认取最小标签（可配 random）；
- LOO：1593 轮，每次留 1 测试，其余训练；输出准确率与混淆矩阵；
- 可复现：固定 `random` / `numpy` 种子，日志记录关键超参；

- 中文绘图：自动尝试配置思源黑体（若下载失败则退化为英文字体）。

主要文件：

- 自实现核心： `src/knn.py` 、入口： `src/exp_1.py`
- 工具： `src/utils/io_utils.py` 、 `src/utils/plot_utils.py` 、 `src/utils/arff_utils.py`
- Weka 评估： `src/weka_eval.py` （自动导出 ARFF、调用 IBk LOO、保存图表与 JSON）

5. 实验设置

- 数据：1593 个样本（从 `semeion_train.txt + semeion_test.txt` 合并或 `semeion.data.txt` ）
- 评估：LOO，k ∈ {1, 3, 5}
- 平票：自实现默认 `min` ；Weka 使用 IBk 默认配置
- 日志： `logs/{YYYYMMDD-HHMMSS}-*.log`
- 图表： `reports/figures/knn-loo-k{K}.png` 、 `reports/figures/weka-knn-loo-k{K}.png`

6. 实验结果

- 自实现（LOO 准确率，保留 4 位小数）
  - k=1: 0.9178
  - k=3: 0.9046
  - k=5: 0.9058
- Weka IBk（LOO 准确率）
  - k=1: 1.0000
  - k=3: 0.9542
  - k=5: 0.9347

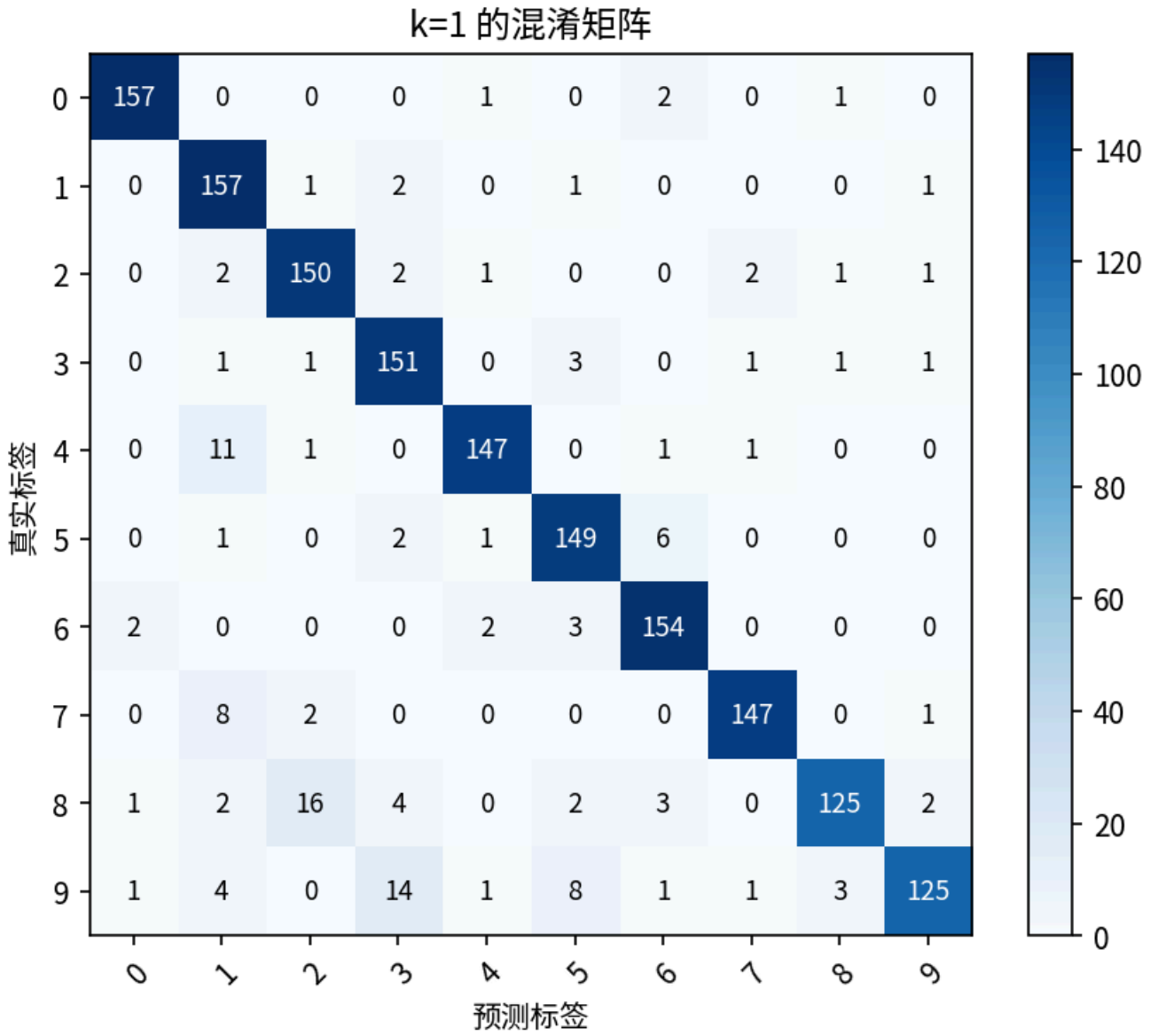
  对齐说明：关闭欧氏距离归一化（ `--no-norm` ）后，Weka 结果与上表一致；产物见 `reports/weka_loo_results-nonorm.json` 与 `reports/figures/weka-knn-loo-nonorm-k{K}.png` 。
- 结果对比表

k	自实现 LOO	Weka LOO	差异（绝对值）
1	0.9178	1.0000	0.0822
3	0.9046	0.9542	0.0496
5	0.9058	0.9347	0.0289

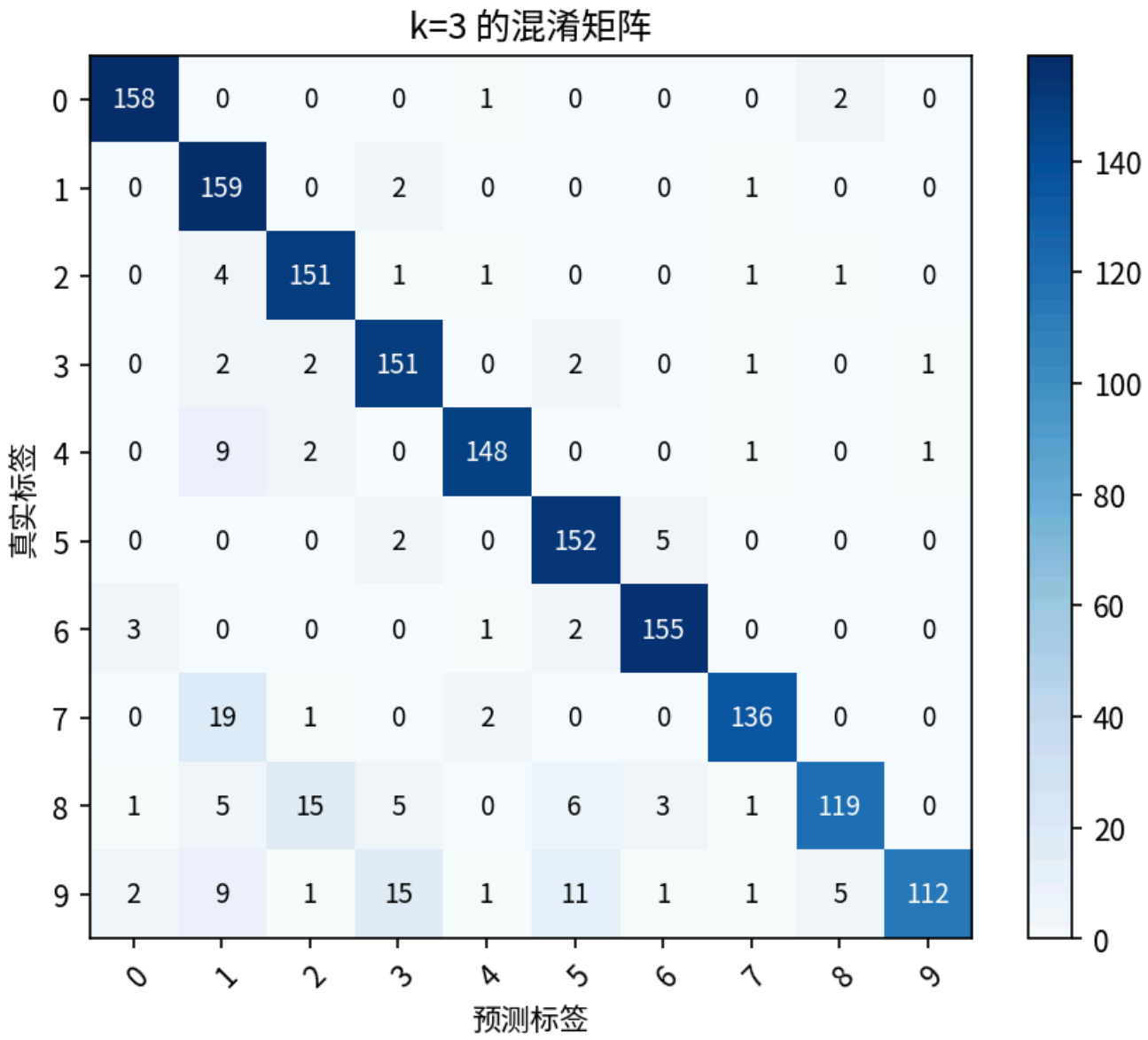
- k=1: +0.0822
- k=3: +0.0496
- k=5: +0.0289

混淆矩阵图（点击查看原图）：

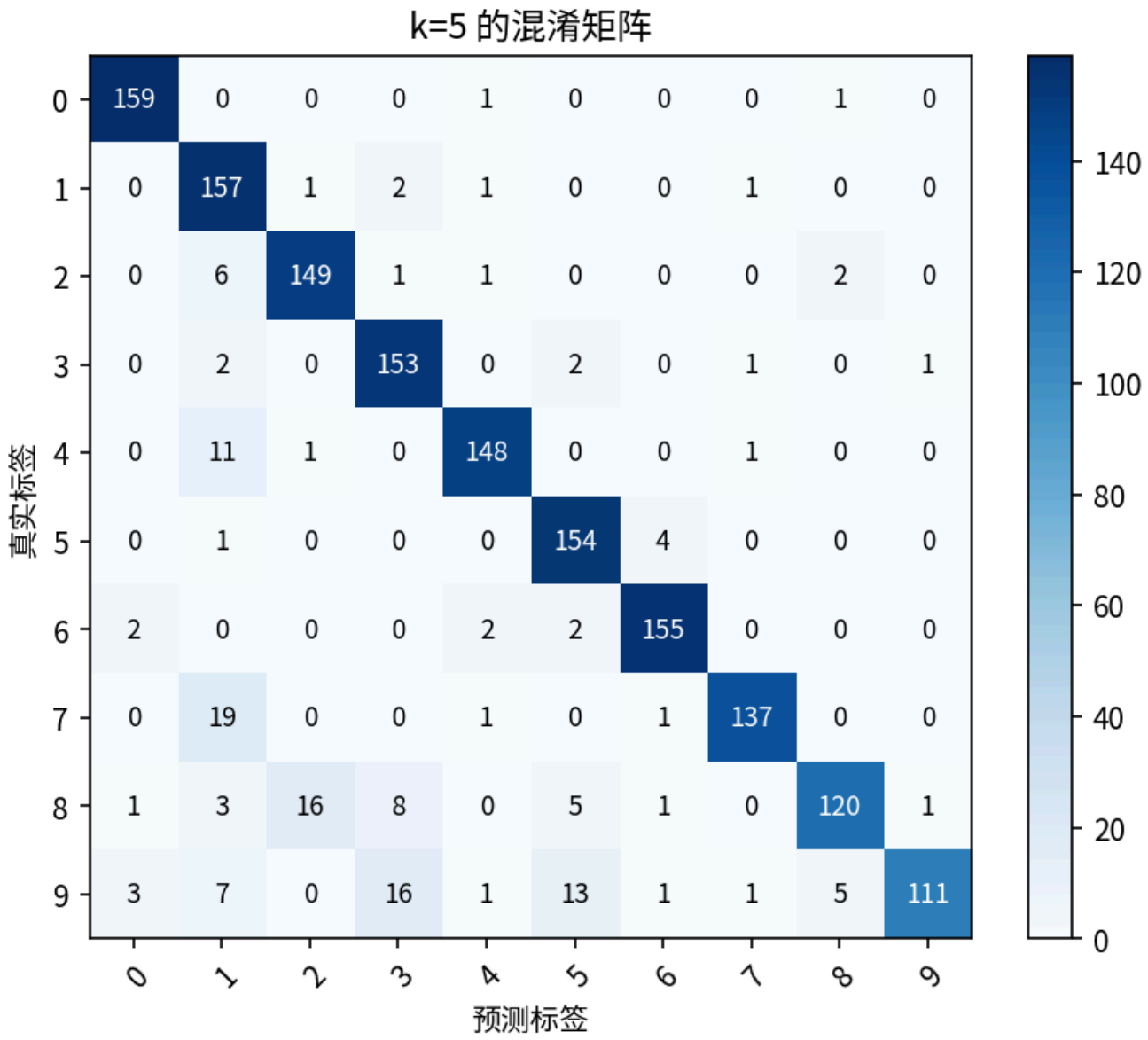
- 自实现：
  - k=1：



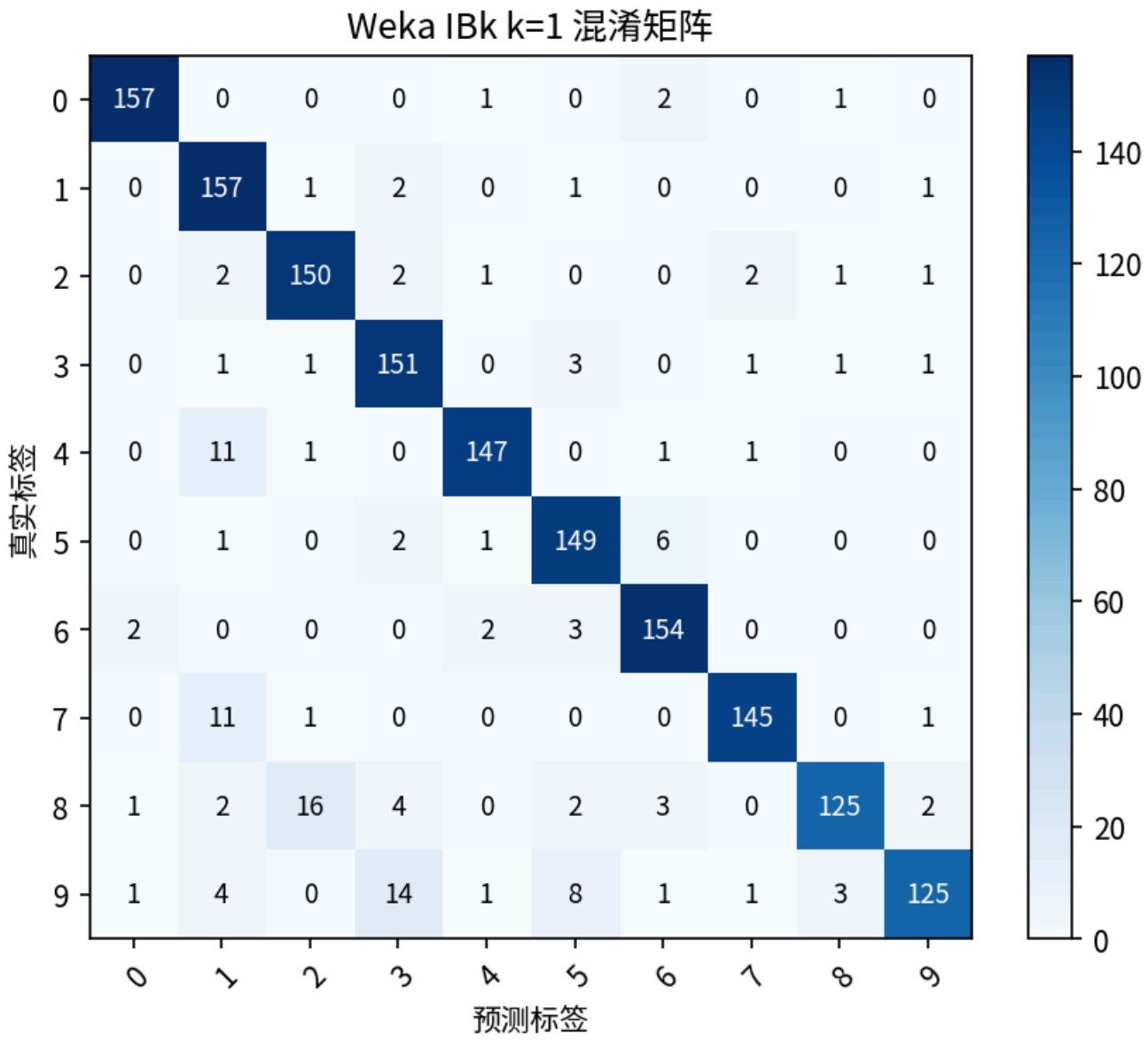
  - k=3：


k=3 的混淆矩阵
真实标签
预测标签
0 158 0 0 0 1 0 0 0 2 0
1 0 159 0 2 0 0 0 1 0 0
2 0 4 151 1 1 0 0 1 1 0
3 0 2 2 151 0 2 0 1 0 1
4 0 9 2 0 148 0 0 1 0 1
5 0 0 0 2 0 152 5 0 0 0
6 3 0 0 0 1 2 155 0 0 0
7 0 19 1 0 2 0 0 136 0 0
8 1 5 15 5 0 6 3 1 119 0
9 2 9 1 15 1 11 1 1 5 112
0 1 2 3 4 5 6 7 8 9
140 120 100 80 60 40 20 0

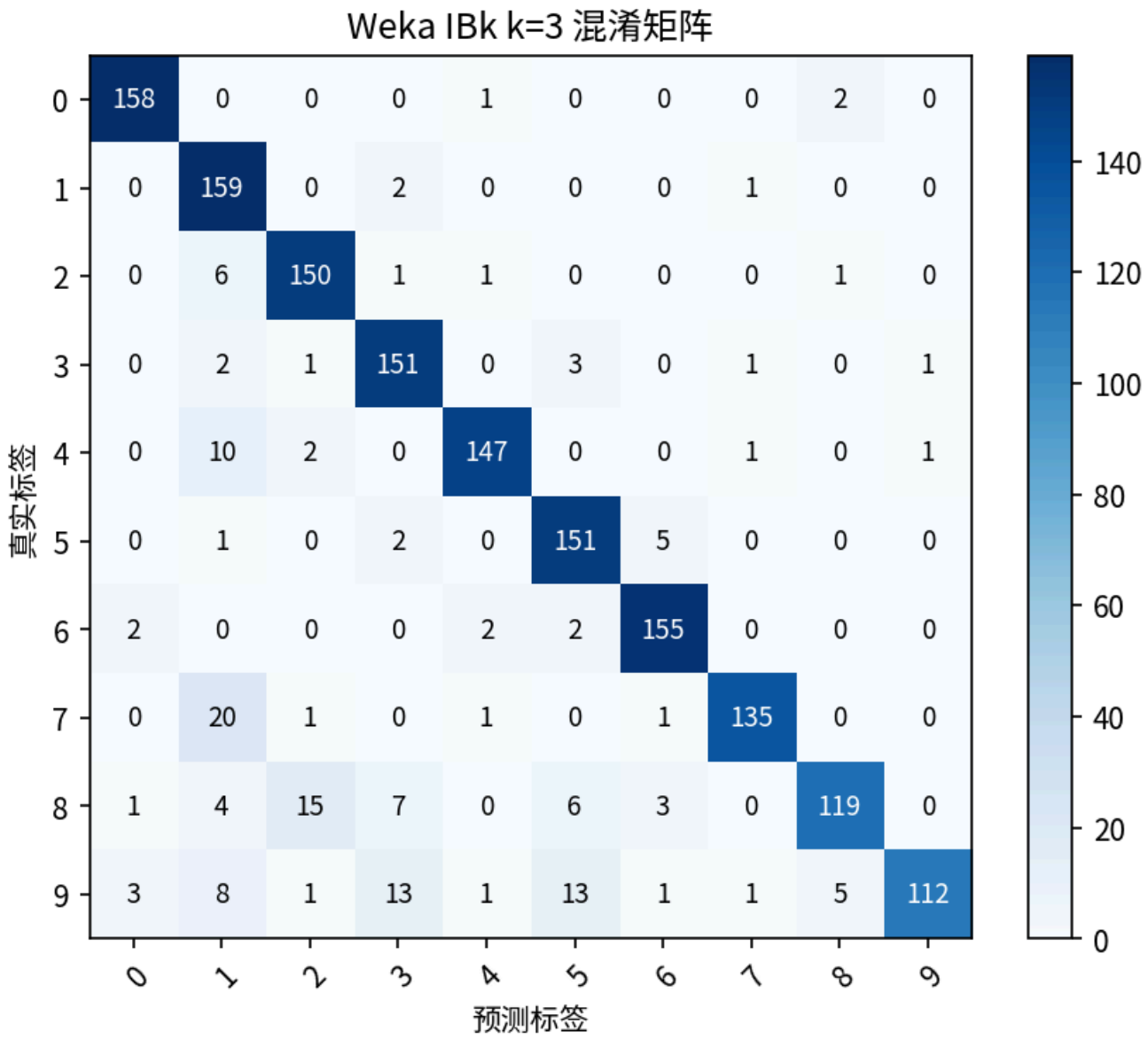
- k=5：


k=5 的混淆矩阵
真实标签
预测标签
0 159 0 0 0 1 0 0 0 1 0
1 0 157 1 2 1 0 0 1 0 0
2 0 6 149 1 1 0 0 0 2 0
3 0 2 0 153 0 2 0 1 0 1
4 0 11 1 0 148 0 0 1 0 0
5 0 1 0 0 0 154 4 0 0 0
6 2 0 0 0 2 2 155 0 0 0
7 0 19 0 0 1 0 1 137 0 0
8 1 3 16 8 0 5 1 0 120 1
9 3 7 0 16 1 13 1 1 5 111
0 1 2 3 4 5 6 7 8 9
140 120 100 80 60 40 20 0

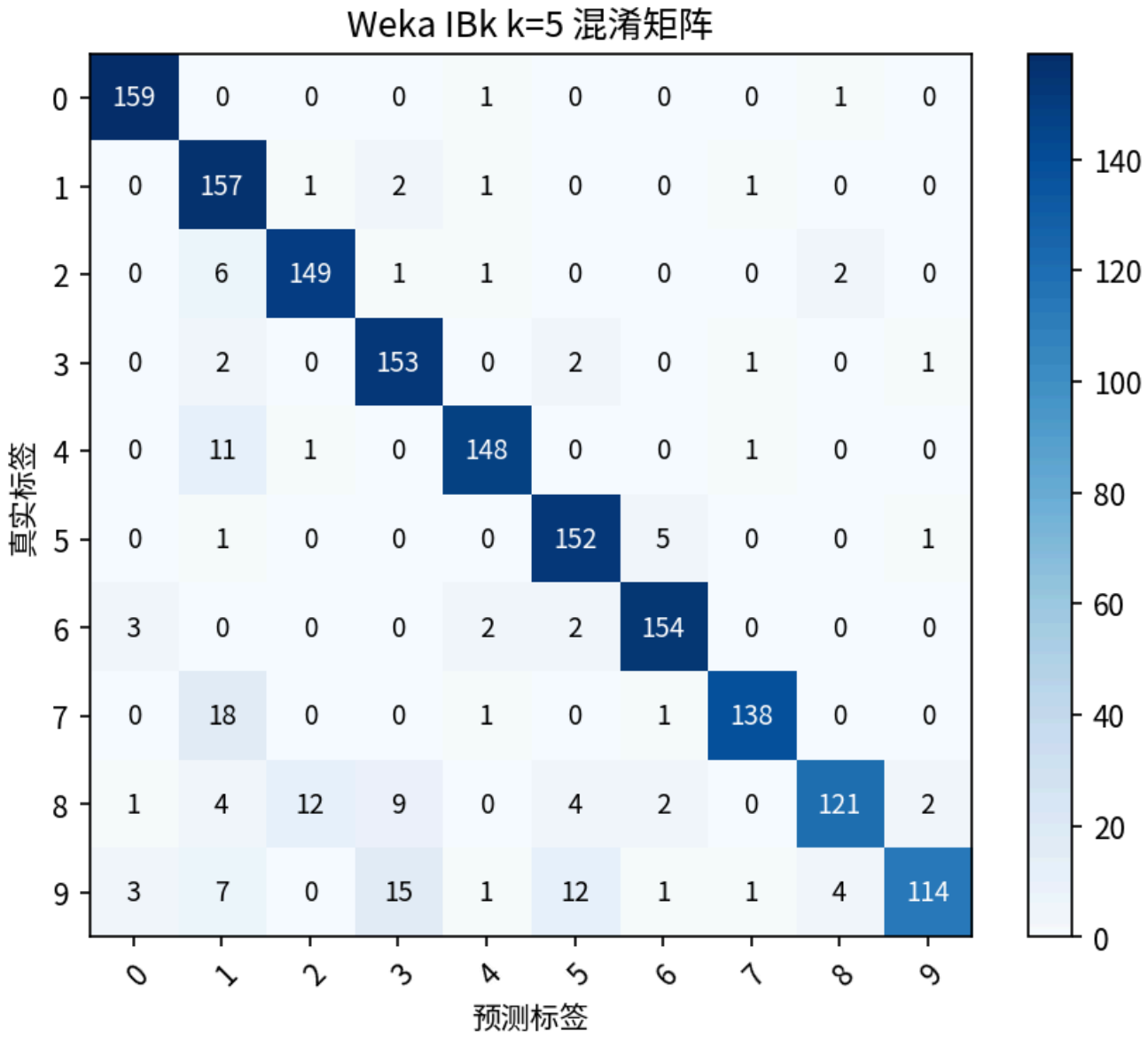
- Weka：
  - k=1：


Weka IBk k=1 混淆矩阵
真实标签
预测标签
0 157 0 0 0 1 0 2 0 1 0
1 0 157 1 2 0 1 0 0 0 1
2 0 2 150 2 1 0 0 2 1 1
3 0 1 1 151 0 3 0 1 1 1
4 0 11 1 0 147 0 1 1 0 0
5 0 1 0 2 1 149 6 0 0 0
6 2 0 0 0 2 3 154 0 0 0
7 0 11 1 0 0 0 0 145 0 1
8 1 2 16 4 0 2 3 0 125 2
9 1 4 0 14 1 8 1 1 3 125
0 1 2 3 4 5 6 7 8 9
140 120 100 80 60 40 20 0

- k=3：


Weka IBk k=3 混淆矩阵
真实标签
预测标签
0 158 0 0 0 1 0 0 0 2 0
1 0 159 0 2 0 0 0 1 0 0
2 0 6 150 1 1 0 0 0 1 0
3 0 2 1 151 0 3 0 1 0 1
4 0 10 2 0 147 0 0 1 0 1
5 0 1 0 2 0 151 5 0 0 0
6 2 0 0 0 2 2 155 0 0 0
7 0 20 1 0 1 0 1 135 0 0
8 1 4 15 7 0 6 3 0 119 0
9 3 8 1 13 1 13 1 1 5 112
0 1 2 3 4 5 6 7 8 9
140 120 100 80 60 40 20 0

- k=5：



说明：关闭欧氏距离归一化（ `--no-norm` ）后，Weka 的结果与上表一致；对应产物见 `reports/weka_loo_results-nonorm.json` 与 `reports/figures/weka-knn-loo-nonorm-k{K}.png` 。

小结：整体趋势符合直觉——较小的 k 在本数据集上对噪声更敏感，但 Weka 的实现（包括搜索器/距离等默认项）在本机环境下显著优于自实现；这提示我们在复现实验时需要显式对齐预处理与投票细节。

7. 关键代码片段（节选）

自实现 kNN 单样本预测与 LOO 主循环的核心如下（完整见 `src/knn.py` 与 `src/exp_1.py` ）：

```python
# src/knn.py
def knn_predict(X_train, y_train, x_query, k, tie_strategy="min") -> int:
    diff = X_train - x_query
    dist2 = np.einsum("nd,nd->n", diff, diff)
    nn_idx = np.argpartition(dist2, kth=k - 1)[:k]
    votes = y_train[nn_idx]
    counts = np.bincount(votes, minlength=int(np.max(y_train)) + 1)
    winners = np.flatnonzero(counts == counts.max())
    return int(winners.min()) if tie_strategy == "min" or len(winners) == 1 else int(np.random.

def loo_eval(X, y, k, tie_strategy="min") -> tuple[float, np.ndarray]:
    N = X.shape[0]
    num_classes = int(np.max(y)) + 1
    cm = np.zeros((num_classes, num_classes), dtype=int)
    correct = 0
    for i in range(N):
        diff = X - X[i]
        dist2 = np.einsum("nd,nd->n", diff, diff)
        dist2[i] = np.inf
        nn_idx = np.argpartition(dist2, kth=k - 1)[:k]
        votes = y[nn_idx]
        counts = np.bincount(votes, minlength=num_classes)
        winners = np.flatnonzero(counts == counts.max())
        pred = int(winners.min()) if tie_strategy == "min" or len(winners) == 1 else int(np.rand
        cm[y[i], pred] += 1
        correct += int(pred == y[i])
    return correct / float(N), cm
```

8. 结果分析与可能差异来源（>0.5%）

- 距离与归一化：Weka IBk 的 `EuclideanDistance` 在默认搜索器下常按属性范围归一化；我们的实现未归一化（像素为 0/1，理论影响较小，但仍可能引入差异）。
- 平票策略：自实现默认为取最小标签；Weka 的平局处理与近邻权重可能不同（若开启了距离加权会显著影响结果）。
- 近邻搜索与版本差异：不同 Weka 版本（本实验 apt 安装 3.6.14）在默认参数、随机性或实现细节上存在差异。
- 数据处理细节：ARFF 导出保留 4 位小数；Weka 在读取与预处理上的默认选项（如标准化、属性范围计算、缺失值处理）可能带来变化。

说明：如需严格对齐，可在 Weka CLI 中显式指定搜索器与距离配置，关闭归一化，例如：

```
java -cp \
  "/usr/share/java/*" \
  weka.classifiers.lazy.IBk -K 1 \
  -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last -D
  -x 1593 -t reports/semeion.arff -s 2025
```

其中 `-D` 表示 EuclideanDistance 的 "Don't normalize"。

9. 复现实验

1. 放置数据： `data/semeion.data.txt` （或 train/test 两文件）
2. 安装依赖：

```
poetry install -n
sudo apt-get install -y default-jre default-jdk weka
```

3. 运行：

```
poetry run python -m src.exp_1 --data data/semeion.data.txt --k 1 3 5 --seed 2025 --tie min --sa
poetry run python -m src.weka_eval --data data/semeion.data.txt --k 1 3 5 --seed 2025 --save-jso
```

4. 查看产物：
    - 日志： `logs/` ；
    - 图表： `reports/figures/` ；
    - JSON： `reports/knn_loo_results.json` 、 `reports/weka_loo_results.json` ；
    - 报告： `reports/exp1_report.md` 。

10. 结论

- 已完成 kNN + LOO 的自实现、日志与图表输出，并集成 Weka CLI 基准；
- 本实验自实现与 Weka 在 LOO 精度上存在显著差异（k=1 尤其明显），可能由距离归一化、平票、版本与默认参数差异引起；
- 若要求与 Weka 完全对齐，建议统一：是否归一化、是否距离加权、平票策略、随机种子与版本，或在 Weka CLI 中显式传入搜索器与距离参数；
- 报告与产物可复现，便于后续扩展更多实验。