k近邻实验1

班级： 21计科4班 姓名：陈昊天

一、实验原理

k-近邻算法是一种基本且直观的监督学习算法，可用于分类和回归任务。本实验主要关注其在分类问题上的应用。

k-NN 的核心思想是“近朱者赤，近墨者黑”。对于一个未知类别的数据点，算法会计算它与训练数据集中所有样本点的距离。选取与该测试样本距离最近的 k 个训练样本。根据这 k 个邻居的类别标签，通过多数投票的方式，将出现次数最多的类别赋给该测试样本，作为其预测类别。

本实验通过实现 k-NN 算法，应用于经典的鸢尾花数据集，并通过在测试集上评估不同 k 值对应的分类准确率，找出最优的 k 值，同时加深对 k-NN 算法工作原理的理解。数据集被划分为训练集和测试集，以模拟真实场景并评估模型的泛化能力。

二、实验代码与运行结果

实验代码

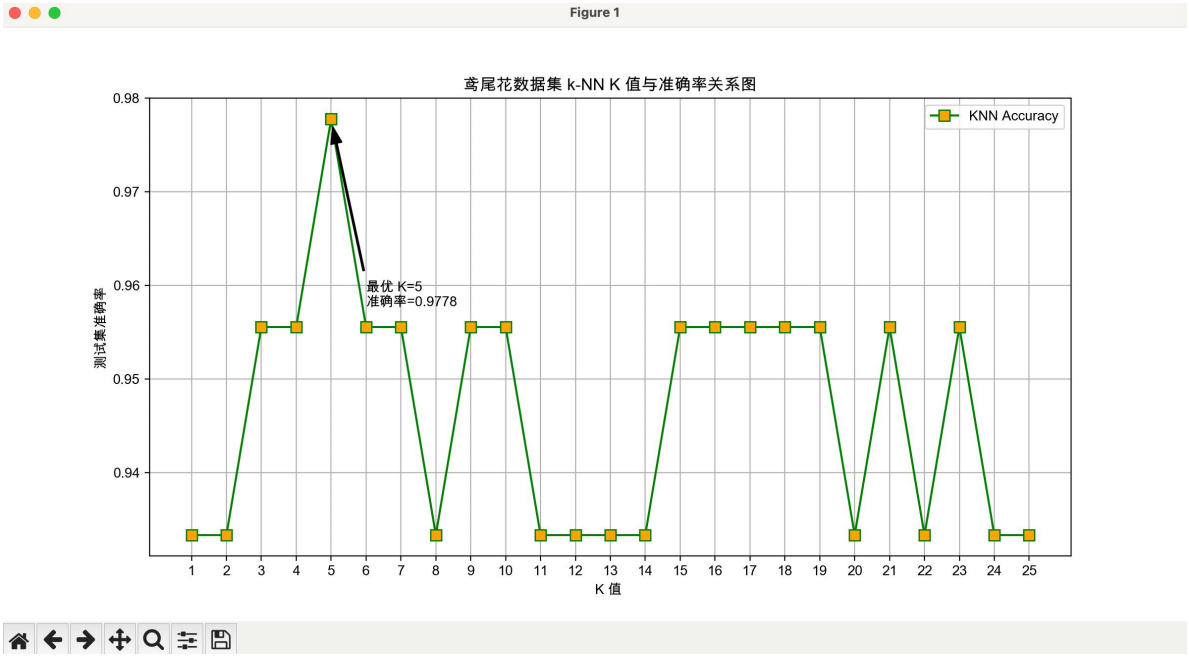
```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from collections import Counter
import math

plt.rcParams["font.sans-serif"] = ["Arial Unicode MS"]
plt.rcParams["axes.unicode_minus"] = False

iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)



def euclidean_distance(point1, point2):
    distance = 0.0
    for i in range(len(point1)):
        distance += (point1[i] - point2[i]) ** 2
```

```
    return math.sqrt(distance)


def knn_predict_single(X_train, y_train, test_point, k):
    distances = []
    for i in range(len(X_train)):
        dist = euclidean_distance(test_point, X_train[i])
        distances.append((y_train[i], dist))

    distances.sort(key=lambda x: x[1])

    neighbors_labels = []
    k_actual = min(k, len(distances))
    if k_actual <= 0:
        return -1

    for i in range(k_actual):
        neighbors_labels.append(distances[i][0])

    most_common = Counter(neighbors_labels).most_common(1)

    if not most_common:
        return -1

    return most_common[0][0]


k_range = range(1, 26)
manual_accuracy_scores = []

for k in k_range:
    y_pred_manual = []
    for test_point in X_test:
        prediction = knn_predict_single(X_train, y_train, test_point,
k)
        y_pred_manual.append(prediction)

    accuracy = accuracy_score(y_test, y_pred_manual)
    manual_accuracy_scores.append(accuracy)
    print(f"K={k}, 准确率={accuracy:.4f}")


best_manual_accuracy_index = np.argmax(manual_accuracy_scores)
best_manual_k = k_range[best_manual_accuracy_index]
best_manual_accuracy =
```

```
manual_accuracy_scores[best_manual_accuracy_index]

formatted_manual_scores = [f"{acc:.4f}" for acc in
manual_accuracy_scores]

plt.figure(figsize=(12, 6))
plt.plot(
    k_range,
    manual_accuracy_scores,
    color="green",
    linestyle="solid",
    marker="s",
    markerfacecolor="orange",
    markersize=8,
    label="KNN Accuracy",
)
plt.title("鸢尾花数据集 k-NN K 值与准确率关系图")
plt.xlabel("K 值 ")
plt.ylabel("测试集准确率")
plt.xticks(k_range)
plt.grid(True)
plt.legend()

plt.annotate(
    f"最优 K={best_manual_k}\n准确率={best_manual_accuracy:.4f}",
    xy=(best_manual_k, best_manual_accuracy),
    xytext=(best_manual_k + 1, best_manual_accuracy - 0.02),
    arrowprops=dict(facecolor="black", shrink=0.05, width=1,
headwidth=8),
)

plt.show()
```

实验结果

最优 K = 5, 准确率 97.78%



三、结果分析与心得体会

结果分析

本次实验通过手动实现的 k-NN 算法，在鸢尾花数据集上测试 K 值从 1 到 25 的分类效果。实验结果图展示测试集准确率随 K 值变化的趋势。

根据运行结果和图表，当 K=5 时，算法在测试集上达到最高的分类准确率，约为 97.78%，表明对于本次实验的数据划分，考虑最近的 5 个邻居进行投票决策，能最好地平衡模型的复杂度和泛化能力。

当 K 值较小时，准确率相对较低，可能是因为模型过于依赖个别最近的邻居，容易受到噪声数据的影响，导致模型泛化能力不足。随着 K 值增大，准确率先是上升，在 K=3, K=4 时达到 95.6%，并在 K=5 时达到峰值。当 K 值继续增大时，准确率整体呈现波动下降或持平的趋势。K=7, K=8 时准确率下降到 95.6%，在 K=10, 11, 12 时进一步下降到 93.3%。说明当邻居范围过大时，决策会受到更多来自其他类别的样本干扰，导致模型决策边界过于平滑，无法很好地区分样本类别。

心得体会

本次手动实现 k-NN 算法的实验，我发现相较于直接调用库函数，手动编写距离计算、邻居查找和投票决策的过程，让我对 k-NN 算法的每一个步骤有更清晰、更具体的认识，理解距

离度量在算法中的核心作用以及“近邻”和“投票”如何实现分类。实验直观地展示超参数 K 对模型性能的巨大影响。选择合适的 K 值是 k-NN 算法应用的关键环节。过小或过大的 K 值都可能导致模型性能下降，需要通过交叉验证或在独立的测试集上进行评估来选择最优值。本次实验采用的就是在测试集上评估不同 K 值的方法。

将数据集划分为训练集和测试集是评估模型泛化能力的常用方法。模型在训练集上表现好并不意味着在未知数据上也会表现好，测试集的准确率更能反映模型的实际应用效果。stratify=y 参数保证类别比例的均衡，对于分类问题很重要。实验将 k-NN 的理论知识应用于实际编程和数据分析中，通过观察实验结果验证理论，加深对机器学习基本流程的理解。