实验报告（1）

学号：202202206341 姓名：王子俊

1. **实验名称： K近邻**
2. **实验目的**

1. 理解K近邻

2. 了解sklearn库

3. 了解pandas库

4. 探究在iris数据集上最优k值与选用不同距离公式的关系

1. **实验内容**

在给定的数据集上，训练K近邻模型，输出预测结果及其准确度。并选择最优k值。更换距离公式并重新测量。

1. **实验步骤**
2. 数据集准备。读取iris数据集。划分测试集和训练集。
3. 模型训练。使用 KNeighborsClassifier
4. 模型预测。使用predict()进行预测，使用sklearn.metrics.accuracy\_score等指标对预测结果进行评估。并画出关于k取值的相关指标变化程度。并选择最优模型。
5. 替换距离公式，改用闵可夫斯基距离和曼哈顿距离重新测量
6. 结果分析
7. **实验结果与分析**

import numpy as np

import matplotlib.pyplot as plt

import sklearn

# 加载数据集

data = load\_iris()

X = data.data

y = data.target

# 划分训练集和测试集，比例 7:3

X\_train, X\_test, y\_train, y\_test = train\_test\_split

(X,y,test\_size=0.3, stratify=y, random\_state=42)

# 涉及比较，必须控制数据集划分变量一致

# 存储不同 k 值的性能指标

k\_values = range(1, 20)  # 将 k 值范围限制为 [1, 20]

precision\_scores = []

recall\_scores = []

f1\_scores = []

accuracy\_scores = []

# 迭代测试不同的 k 值

for k in k\_values:

knn = KNeighborsClassifier(n\_neighbors=k##,metric='manhattan', metric='minkowski')

# 改用不同距离公式

knn.fit(X\_train, y\_train)

y\_pred = knn.predict(X\_test)

# 计算查准率、召回率、F1 和准确率

 precision\_scores.append(precision\_score(y\_test,y\_pred,average='macro'))

recall\_scores.append(recall\_score(y\_test, y\_pred,average='macro'))

f1\_scores.append(f1\_score(y\_test, y\_pred, average='macro'))

accuracy\_scores.append(accuracy\_score(y\_test, y\_pred))

# 绘制不同 k 值的性能指标图

plt.figure(figsize=(12, 8))

plt.plot(k\_values, precision\_scores, label="Precision",marker='o')

plt.plot(k\_values, recall\_scores, label="Recall", marker='o')

plt.plot(k\_values, f1\_scores, label="F1 Score", marker='o')

plt.plot(k\_values, accuracy\_scores, label="Accuracy", marker='o')

plt.xlabel("Number of Neighbors (k)")

plt.ylabel("Score")

plt.title("KNN Performance Metrics for Different k Values")

plt.legend()

plt.grid()

# 寻找肘点（最大F1分数对应的k值）

best\_k = k\_values[np.argmax(f1\_scores)]

best\_f1 = max(f1\_scores)

print(f"最佳的 k 值: {best\_k}，对应的 F1 分数: {best\_f1:.2f}")

# 再训练模型并显示最终结果

knn\_best = KNeighborsClassifier(n\_neighbors=best\_k)

knn\_best.fit(X\_train, y\_train)

y\_pred\_best = knn\_best.predict(X\_test)

# 输出最终的查准率、召回率、F1分数和准确率

print("最终模型性能:")

print(f"Precision:{precision\_score(y\_test,y\_pred\_best,average='macro'):.2f}")

print(f"Recall:{recall\_score(y\_test,y\_pred\_best，average='macro'):.2f}")

print(f"F1Score:{f1\_score(y\_test,y\_pred\_best,average='macro'):.2f}")

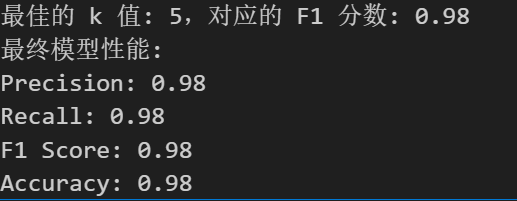
print(f"Accuracy: {accuracy\_score(y\_test, y\_pred\_best):.2f}")

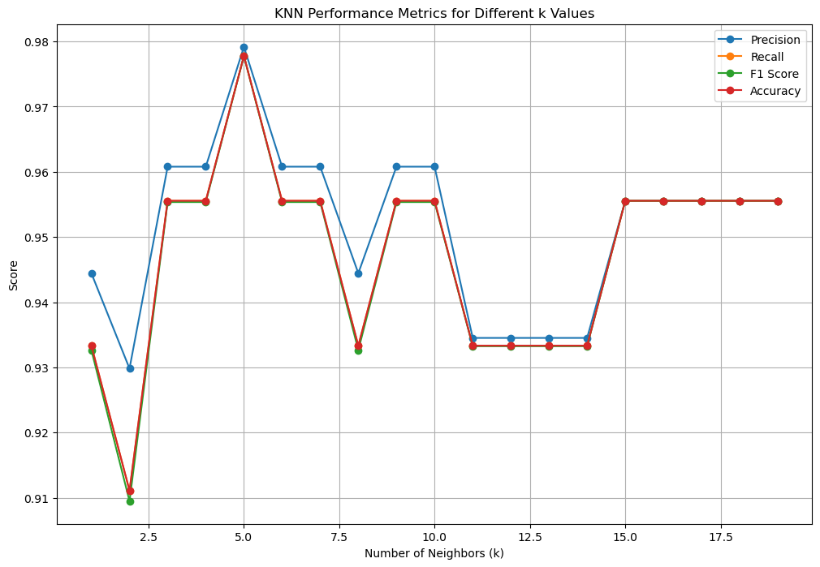
# 显示图形

plt.show()

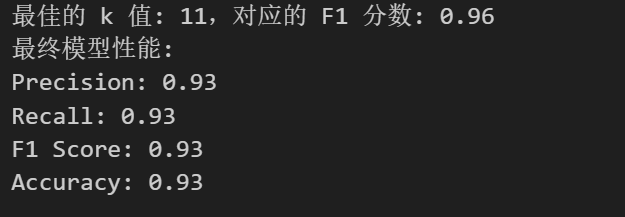
**实验结果**：

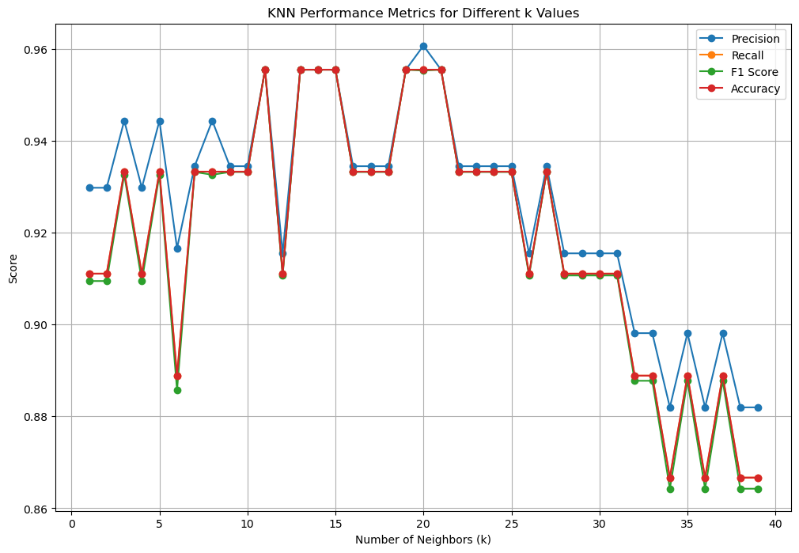
Euclidean Distance：





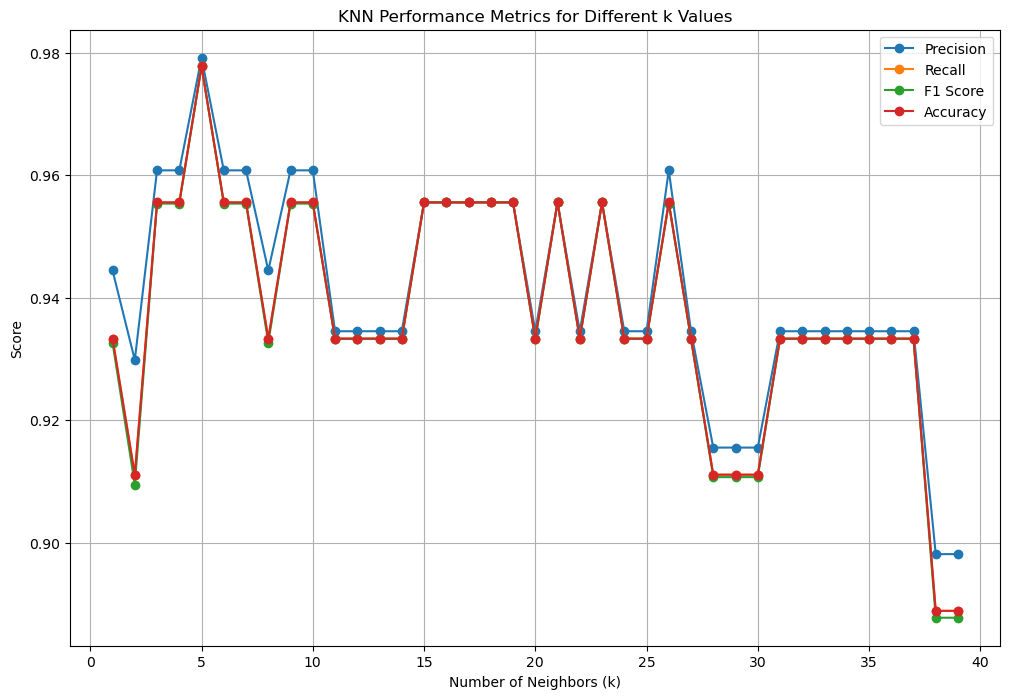
Manhattan Distance





Minkowski Distance：





**实验分析**（分析图表中的结果如何、分析算法的有效性、效率等）

如图表中所示，可得出以下结论：

1. 欧式距离和闵可夫斯基距离基础下四个评判标准均在k=5处取得最大值，说明在此数据集上k=5分类情况最好。但是在曼哈顿距离下呈现出k=11时效果最佳。
2. 未防止过拟合，本次实验仅选择40以下的k值进行试验，发现均在38处有，明显下降
3. 最优模型指标F1值均在0.95左右，说明分类误差较小，模型效果较好
4. **实验总结**

KNN（K-Nearest Neighbors）算法是一种基于实例的学习方法，具有较为简单的实现过程，无需提前进行训练。这使得它在一些应用中非常便捷，尤其适合于小规模数据集上，能通过距离度量在分类任务中取得显著效果。然而，KNN算法也存在一定的局限性，特别是在面对大规模数据集时，由于需要在每次预测时计算与所有训练样本的距离，因此计算量大，导致预测速度较慢，尤其在数据集庞大时，性能表现不佳。此外，KNN对距离度量公式的选择有较高的要求，不同的距离度量（如欧氏距离、曼哈顿距离、余弦相似度等）在不同问题中可能表现出不同的效果，这要求我们深入理解距离的内在意义及其对模型结果的影响，选择合适的度量方式。