实验

实验要求：

用 k- NN 算法对鸢尾花数据集进行分类。

k-NN 算法具体可网上查询具体算法进行学习。

数据：iris 数据集包含在 sklearn 库当中，具体在 sklearn\datasets\data 文件夹下，文件名为 iris.csv。也可自行网上下载。

环境：python 3

验收方式：实验报告

报告书写要求：

一、k-NN 算法

K 最邻近（KNN，K-Nearest Neighbor）分类算法是数据挖掘分类技术中最简单的方法之一。所谓 K 最近邻，就是 K 个最近的邻居的意思，说的是每个样本都可以用它最接近的 K 个邻近值来代表。近邻算法就是将数据集合中每一个记录进行分类的方法

KNN 算法的核心思想是，如果一个样本在特征空间中的 K 个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。该方法在确定分类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。KNN 方法在类别决策时，只与极少量的相邻样本有关。由于 KNN 方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于类域的交叉或重叠较多的待分样本集来说，KNN 方法较其他方法更为适合。

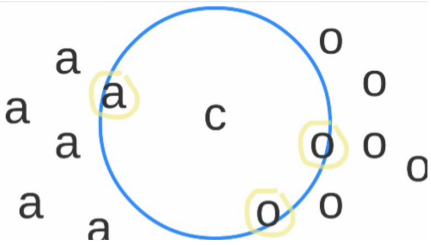

图 1 k-NN 算法示意图

如图 1 所示，可分为 *a*, *o* 两类，对于不明确的 *c* 来说，根据给定的 *k* 作为距离，在范围内相对更多的类别将作为 *c* 的类别，图中，*c* 就属于 *o* 类。

二、程序实现

首先对数据集进行划分，划分为训练集和测试集。训练集随机选择其中的 15 组数据，而测试集则选择剩余的 135 组数据。

需要注意的是，这里的数据在处理发现存在部分问题，即：sklearn 库中的数据来源为 UCI 官方的鸢尾花数据集，但查询 UCI 官方后发现其数据集和 sklearn 库中的数据集并不一致，因此这里使用 UCI 官方的数据集。

数据集地址: http://archive.ics.uci.edu/ml/datasets/Iris

```
import numpy as np
# 划分数据集
def DivideDataset():
    index = (np.random.permutation(len(iris)))[0:15]
    iris_test  = iris.take(index)
    iris_train = iris.drop(index)
    datasets = [iris_test, iris_train]
    return datasets, iris_test, iris_train
```

读入和得到的数据集如图 2 所示：

	sepal_len	sepal_width	petal_len	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

图 2 鸢尾花数据集

K-NN 算法的实现在理解其原理后也十分自然，无非就是对测试数据和训练数据进行逐一的距离计算，得到结果。

```
from collections import Counter
def KNNAlgorithm(train, test, Class_Name, k):
    difference = np.tile(test, (train.shape[0], 1)) - train
    distance = (difference**2)**0.5
    # 消除列，计算每一行的 sum
    results = distance.sum(axis=1)
    # 从小到大排序，返回索引值
```

```
    results = results.argsort()

    label = []
    for i in range(k):
        label.append(Class_Name[np.where(results==i)][0])

    collection = Counter(label)
    final = collection.most_common(1)
    return final
```

三、分类结果

分类结果对于 k 的选择是有关联的，但同一组数据之下，一般不会有过大的差异，图 3 展示了某一组数据下的分类结果和原标签对比，可以看出基本一致，仅有少量数据分类错误，准确率在 15 组测试数据下，达到 86.667%。可以预见的是，在更多训练数据的支持下，准确度会更高。

```
----------------
Iris-setosa
Iris-setosa
----------------
Iris-setosa
Iris-setosa
----------------
Iris-versicolor
Iris-versicolor
----------------
Iris-versicolor
Iris-versicolor
----------------
Iris-setosa
Iris-setosa
----------------
Iris-versicolor
Iris-virginica
----------------
Iris-setosa
Iris-setosa
Iris-setosa
Iris-setosa
----------------
Iris-setosa
Iris-setosa
----------------
Iris-setosa
Iris-setosa
----------------
Iris-virginica
Iris-virginica
----------------
Iris-virginica
Iris-virginica
----------------
Iris-virginica
Iris-virginica
----------------
Iris-virginica
Iris-virginica
----------------
86.66666666666667
```

图 3 测试结果

四、收获与体会

本实验中，学习到了机器学习分类算法中较为基础的 k-近邻（k-NN）算法，应用了课程中学到的 Python 数据分析常用的语法和工具，对算法原理有了更深的理解。同时对算法实现有了初步的尝试，并且使用该算法实现了对鸢尾花数据集分类问题的测试。过程中不免遇到一些原理上、语法上的问题，但都通过查阅相关文献以及官方 Document 得到了解决。