计算机学院 数据科学概论 课程实验报告

实验题目：Knn 算法实现		学号：202200130041
日期：2023/4/27	班级： 22 级数据	姓名：左景萱
Email：zuojingxuan1130@mail.sdu.edu.cn		

实验目的：

Knn 是一种简单、但是功能强大的分类算法，可以实现复杂的分类效果。Knn 是一种有监督的机器学习方法。

本实验首先安装配置好 Anaconda 以及 Python 环境，通过在 iris 数据集上用 Knn 算法实现分类操作，从而使同学们熟练掌握 Knn 算法的原理并学会将其运用到实践中。

实验软件和硬件环境：

1. 操作系统：Windows10；
2. Anaconda 版本：5.3.0；
3. Python 版本：3.6.8；

实验步骤与内容：

一. 根据给出的实验实例进行 KNN 算法的学习，了解 scikit-learn 中 KNN 算法实现的的简明用法，以及如何选择合适的 K 值。

二. 完成 KNNClassifier 的填空

（1）了解各个函数的功能。

```
def __init__(self,k):
    assert k>=1,"k must be valid"
    self.k = k
    self._X_train = None
    self._y_train = None
```

在实例化类对象的时候初始化 k 的值并检查 k 值得合法性。

```
def fit(self, X_train, y_train):
    """根据训练数据集X_train和y_train训练kNN分类器"""
    assert  X_train.shape[0] == y_train.shape[0], \
        "the size of X_train must be equal to the size of y_train"
    assert self.k <= X_train.shape[0], \
        "the size of X_train must be at least k"

    self._X_train = X_train
    self._y_train = y_train
    return self
```

Fit 方法将训练集的 X，y 导入并且检查结果的合法性。

```python
def predict(self, X_predict):
    """给定待预测数据集X_predict, 返回表示X_predict的结果向量"""
    assert self._X_train is not None and self._y_train is not None, \
        "must fit before predict!"
    assert X_predict.shape[1] == self._X_train.shape[1], \
        "the feature number of X_predict must be equal to X_train"

    y_predict = [self._predict(x) for x in X_predict]
    return np.array(y_predict)
```

Predict 方法检查预测数据的合法性，并且遍历 test 集中每一个数据，调用 _predict 方法返回对应的分类答案加入到 y_predict 中，最后返回 y_predict。

```python
def _predict(self, x):
    """给定单个待预测数据x,返回x的预测结果值"""
    assert x.shape[0] == self._X_train.shape[1], \
        "the feature number of x must be equal to X_train"
    x=np.array(x).reshape((1,-1))
    distances = [sqrt(np.sum((x_train - x) ** 2)) for x_train in self._X_train]
    nearest_K = np.argsort(distances)
    top_k = [self._y_train[i] for i in nearest_K[:self.k]]
    count = Counter(top_k)
    y_predict = count.most_common(1)[0][0]
    return y_predict
```

_predict 方法中其实不必检查 x 的合法性，因为 predict 方法中的检查已经保证了传入数据都是合法的。然后将 x 进行 reshape 以在下一步可以进行广播操作。然后利用广播机制算出 x 距离每一个 train 数据的 Euler 距离，利用 argsort 方法取出距离最近的 k 个点的 index，将对应的类别加入到 top_k 当中。最后用 Counter 中的 most_common 方法取出在 k 个点中出现最多的类别并且返回作为预测类别。

（2）进行实现与效果比对。

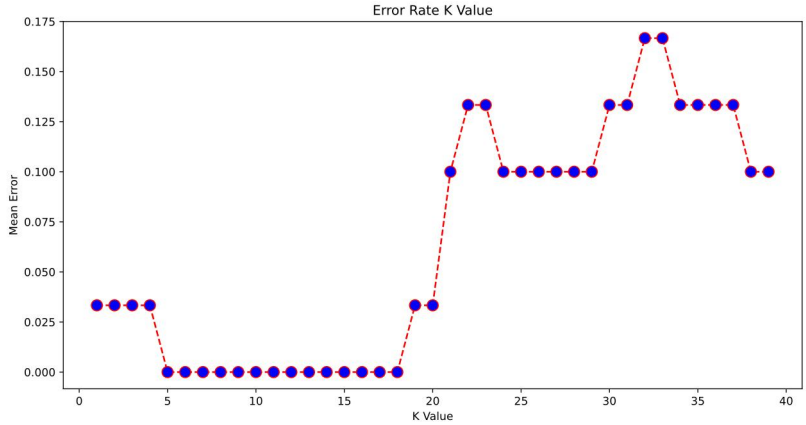
首先需要设置 random_state=0 以保证得到的答案与实例中一致。然后 k=5 的答案如下：

```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

是正确的。在 range（1，40）中的图像如下：



发现和实例中是一样的，所以是正确的。

结论分析与体会：

1. KNN 算法的原理简单但是效果在选择了合适的 K 的情况下却很好。

2. KNN 算法适用于中小型数据集，因为它需要计算待分类样本与所有训练集样本之间的距离，并选择 K 个最近邻。对于大型数据集，KNN 算法的计算复杂度会很高，需要采用一些优化方法，如 KD 树、球树等来加速计算。

3. KNN 算法对数据分布的敏感度比较高，它假设相似的样本在空间上彼此接近。如果数据分布不均匀或存在离群点等情况，KNN 算法的性能会受到影响。因此，在使用 KNN 算法时，需要考虑数据分布和异常值等问题，采取相应的处理方法，如去除离群点、降维、集成学习等以保证算法的可靠性。

就实验过程中遇到的问题及解决处理方法，自拟 1－3 道问答题：

1. 为什么要进行 normalization 规范化预处理？

答：KNN 算法是一种基于距离度量的分类算法，它通过计算待分类样本与训练集中各个样本之间的距离来进行分类。由于不同特征的取值范围和单位不同，这会导致距离计算的结果受到不同特征的影响程度不同，进而影响分类结果的准确性。因此，在使用 KNN 算法时，需要对数据进行规范化处理，将不同特征的取值范围和单位转换为统一的尺度，以消除不同特征之间的量纲差异和偏差，从而提高模型的准确性和泛化能力。