山东大学

信息科学与工程学院

2022－2023 学年第一学期

实 验 报 告

课程名称：信息基础 II

专 业 班 级 崇新学堂

学 生 学 号

学 生 姓 名

课 程 报 告 KNN 最近邻算法

1. 实验目的

在鸢尾花数据集的基础上测试 KNN 最近邻算法。

2. 实验要求

不能直接使用 ML 库或开源代码

3. 实验过程

3.1 数据处理

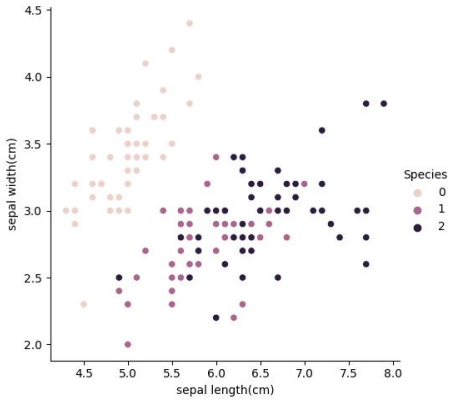
导入鸢尾花数据集，并且按照 8：2 的比例划分训练集和验证集。

```
def DataSet():
    iris = datasets.load_iris()
    x_tr, x_te, y_tr, y_te =skmodel.train_test_split(iris.data, iris.target, test_size=0.2) #划分数据集，验证集占20%
    x_train = np.array(x_tr)    #转化为array格式
    x_test = np.array(x_te)
    y_train = np.array(y_tr)
    y_test = np.array(y_te)
    return x_train, x_test, y_train, y_test
```

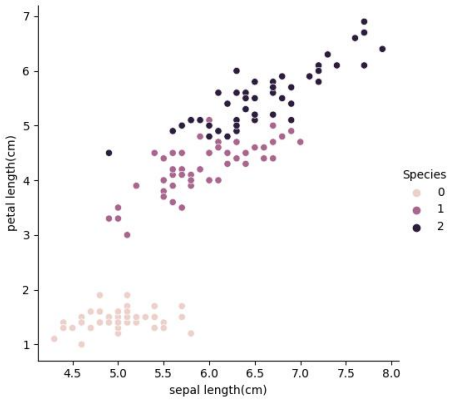
3.2 数据可视化

在进行 KNN 算法之前，先将数据进行可视化，将特征两两组合，绘制出特征之间的二维平面图，对数据有一个直观的判断。

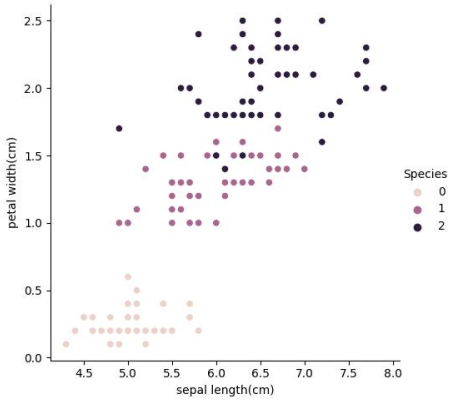
```
def View():
    iris = datasets.load_iris()
    iris_df = pd.DataFrame(iris['data'], columns=   #将数据转化为DF形式用于绘图
                           ['sepal length(cm)',
                            'sepal width(cm)',
                            'petal length(cm)',
                            'petal width(cm)'])
    iris_df['Species'] = iris.target
    sns.relplot(x='petal length(cm)', y='petal width(cm)', data=iris_df, hue='Species')
    plt.show() #绘制在二维平面内的六张图片
```



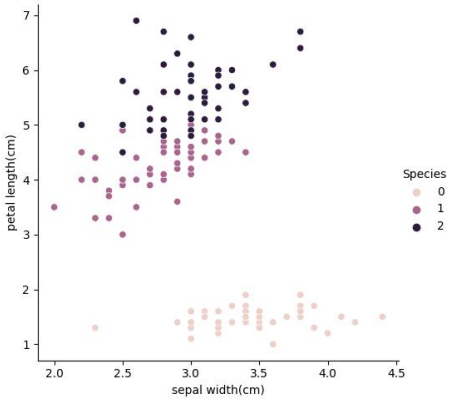
a) Sepal length-sepal width



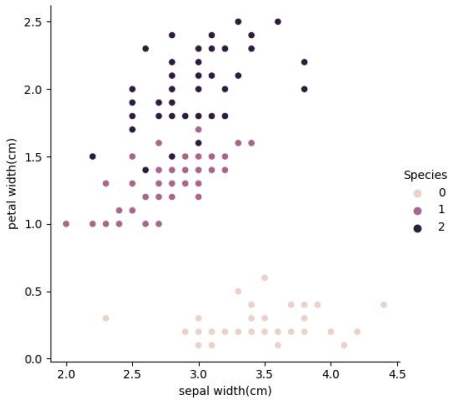
b) Sepal length-petal length



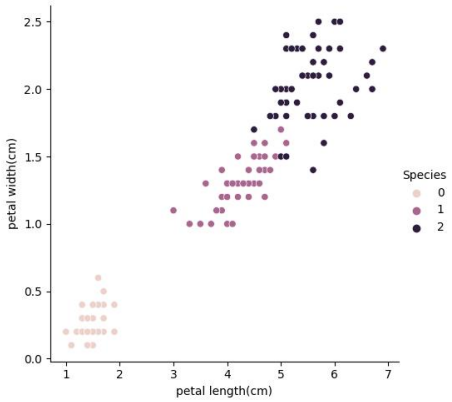
c) Sepal length-petal width



d) Sepal width-petal length



e) Sepal width-petal width



f) Petal length-petal width

3.3 KNN 算法

KNN 的算法比较简单，选取距离预测点最近的 K 个点，之后分类统计 K 个点，个数最多的种类就是最终的预测结果。代码如下：

```python
def KNN(x_predict, x, label, k):
    A = 0
    B = []
    c = 0
    x_size = x.shape[0]          #求取数据行数，也就是求取训练集的个数
    distance = (np.tile(x_predict, (x_size, 1)) - x) ** 2 #tile函数的作用是将待预测输入列表进行纵向复制，从而求取距离平方
    sum_distance = distance.sum(axis=1)      #对获得的距离平方进行求和
    sqrt_distance = sum_distance ** 0.5      #开方获得欧氏距离（这一步可省略）
    order = sqrt_distance.argsort()          #随后对欧氏距离进行排序，按从小到大的顺序排序，而返回值为排序前的index
    neighbor_set = {}                        #建立一个字典，后续使用
    for i in range(k):
        neighbor_point = label[order[i]]     #统计前k个点中，不同种类的点的个数
        neighbor_set[neighbor_point] = neighbor_set.get(neighbor_point, 0) + 1
    for j in range(len(neighbor_set.keys())):    #根据数量最多的种类进行判决
        B = list(neighbor_set.keys())
        if neighbor_set[B[j]] > A:
            A = neighbor_set[B[j]]
            c = B[j]
    return c
```

在 KNN 算法中，距离选择欧式距离，获取各点距离预测点的距离之后，对距离进行排序，选取距离最近的前 k 个点，随后根据各类点的个数进行判决，从而得出分类，之后利用写好的 KNN 算法进行预测，验证其准确率，通过多次反复实验，寻求最佳的 k 值。

```python
right_number_train = 0
right_number_test = 0
x_train, x_test, y_train, y_test = DataSet()
number_test = x_test.shape[0]
number_train = x_train.shape[0]
label = []
print(x_test[0])
for i in range(number_test):          #计算验证集的分类正确率
    a = KNN(x_test[i], x_train, y_train, 1)
    if a==y_test[i]:
        right_number_test=right_number_test+1
print('验证集分类正确率为：',right_number_test/number_test)

for i in range(number_train):         #计算训练集的正确率
    a = KNN(x_train[i], x_train, y_train, 1)
    if a==y_train[i]:
        right_number_train=right_number_train+1
print(('训练集分类正确率为：',right_number_train/number_train))
rate=(right_number_test+right_number_train)/(number_test+number_train)
print('分类正确率为：',rate)
```

K 值与正确率的关系如下图所示，最终发现点数在 15 左右的分类效果是比较好的。