第一次竞赛

一、问题解决过程：

采用 k 近邻算法

Step1:读取训练集数据，得到训练集数据的属性 data_property 和训练集数据的类别 data_class

Step2：采用 knn 算法进行训练模型（kNN 函数）

①计算测试数据到训练集各个数据的距离

②对距离进行排序，得到排好序后元素在原位置的索引

③选择 k 个近邻

④计算 k 个近邻中出现各个类别的次数

⑤返回出现次数最多的类别标签，即预测的类别

Step3：读取测试集数据的属性 test_data

Step4：对测试集中的每一个样本数据用以上模型进行预测得到预测类别

Step5：将预测的类别按照指定的格式保存到 csv 文件中

二、难点解决过程：

①近邻数 k 的取值，对于不同的 k 得到的预测准确率是不同的，通过将训练集中的一部分数据拿来训练，另一部分数据用来预测，得到准确率较高时的 k，再用测试集多次测试得到 k=15 时得到的预测结果是最佳的

②起初预测结果的准确率并不高，因为只拿了 80%的训练集进行了训练，忽略了剩余的 20%的训练集数据，导致测试集预测的准确率不高。在用全部的训练集进行训练之后测试集预测的准确率上升

三、核心代码分析：

```
def kNN(test,train_property,train_class,k):
```

```
    # 测试数据    训练集属性 训练集类别   近邻数
    #   #step1 计算测试样本到训练集每一个样本的距离
    m,n=np.shape(train_property)#训练集的行数、列数
    difference=np.tile(test,(m,1))-train_property#按元素求差值
    #np.title(A,reps):构造一个矩阵，将 A 重复 reps 次
    #将 test 行重复 m 次，列重复一次
    sqd=difference**2#将差值平方
    numSqd=np.sum(sqd,axis=1)#按行累加
    distance=numSqd**0.5#开平方得到距离
    #   #step2  对距离进行排序
    distanceIndex=np.argsort(distance)#返回排序后其中元素在原来位置的索引
    #print(distanceIndex)
    classCount={}
    for i in np.arange(k):
        #step3 选择 k 个近邻
        #distanceIndex[i]  例如当 k=0 时，即找出 distanceIndex 中的第一个值，第一个值是排好序后最小的距离（在原来位置）的索引
        #train_class[distanceIndex[i]]  :  由 原 来 位 置 的 索 引（distanceIndex[i]）就能得到对应的类别
        classNumber=train_class[distanceIndex[i]]#  有  了  索  引 distanceIndex[i]就能找到相应位置的类别
```

```
            #step4 ： 计 算 k 个 最 近 邻 中 出 现 各 个 类 别 的 次 数
classCount[classNumber]=classCount.get(classNumber,0)+1
        #   #step 5： 返回出现次数最多的类别标签，即预测的类别
        maxCount=0
        for key,value in classCount.items():
            if value>maxCount:
                maxCount=value
                maxIndex=key
        return maxIndex
```