一、**KNN（K近邻算法）：可以做分类，也可以做回归**

　KNN算法我们主要要考虑三个重要的要素，对于固定的训练集，只要这三点确定了，算法的预测方式也就决定了。这三个最终的要素是k值的选取，距离度量的方式和分类决策规则。

KNN算法的实现方式：蛮力实现、KD树实现、球树实现、BBF树实现、MVP树实现等

大致思路：一个样本在特征空间中K个最相似的样本中大多数属于那个类，则该样本就属于这个类；

二、算法生成流程：

1、计算测试数据点与各个训练数据之间的距离；

2、按照距离的递增关系进行排序；

3、选取距离最小的K个点；

4、确定K个点中所有类别出现的频率；

5、返回前K个点中出现频率最高的类别，作为测试数据点的预测分类。

三、优缺点：

优

1、简单，容易实现

2、特别适合多分类的问题

3、可用于非线性分类

4、对异常值不敏感

5、适合样本容量较大的分类

缺

1、不适合用于样本不平衡的情况，

2、计算量较大，

3、KD数，球树之类的模型很费内存；

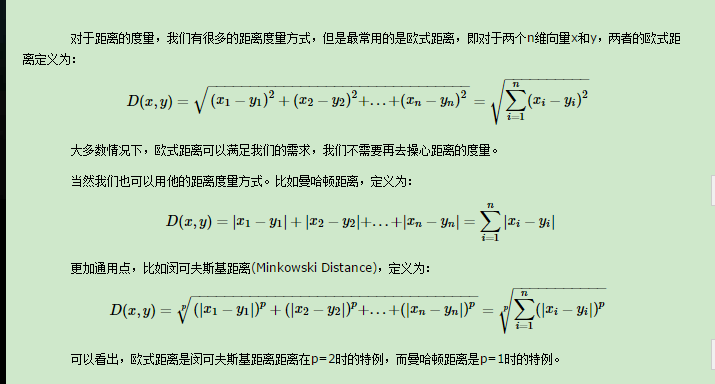
四、K值的优化

选择较小的K值时，训练误差会减少，但是带来的问题是泛化误差增大，换句话说，K值得减少就意味着模型变得复杂了，容易发生过拟合；

选择较小的K值时，训练误差会增多，但是带来的问题是泛化误差减少，换句话说，K值得减少就意味着模型变得简单了，容易发生欠拟合；

对于K值的选择，没有一个固定的经验，一般根据样本的分布，通过交叉验证选择一个合适的K值。

五、距离度量的方式



1. KNN中KD树的实现：

我们首先来看建树的方法。KD树建树采用的是从m个样本的n维特征中，分别计算n个特征的取值的方差，用方差最大的第k维特征Nk来作为根节点。对于这个特征，我们选择特征Nk的取值的中位数Nkv对应的样本作为划分点，对于所有第k维特征的取值小于Nkv的样本，我们划入左子树，对于第k维特征的取值大于等于Nkv的样本，我们划入右子树，对于左子树和右子树，我们采用和刚才同样的办法来找方差最大的特征来做更节点，递归的生成KD树。

## **KD树搜索最近邻**

当我们生成KD树以后，就可以去预测测试集里面的样本目标点了。对于一个目标点，我们首先在KD树里面找到包含目标点的叶子节点。以目标点为圆心，以目标点到叶子节点样本实例的距离为半径，得到一个超球体，最近邻的点一定在这个超球体内部。然后返回叶子节点的父节点，检查另一个子节点包含的超矩形体是否和超球体相交，如果相交就到这个子节点寻找是否有更加近的近邻,有的话就更新最近邻。如果不相交那就简单了，我们直接返回父节点的父节点，在另一个子树继续搜索最近邻。当回溯到根节点时，算法结束，此时保存的最近邻节点就是最终的最近邻。

## **KD树预测**

有了KD树搜索最近邻的办法，KD树的预测就很简单了，在KD树搜索最近邻的基础上，我们选择到了第一个最近邻样本，就把它置为已选。在第二轮中，我们忽略置为已选的样本，重新选择最近邻，这样跑k次，就得到了目标的K个最近邻，然后根据多数表决法，如果是KNN分类，预测为K个最近邻里面有最多类别数的类别。如果是KNN回归，用K个最近邻样本输出的平均值作为回归预测值。