# 1.vs直接打开项目。

# 2.数据分析与处理：

数据装载：读取excel表格内容；数据清洗：去除相同行；数据归一化：将属性值用数字表示。

# 3.技术和创新：介绍KNN算法

## 3.1距离测量

在同一个样本空间内，测量两个点之间的距离，如果两个点的距离越近，那就说明两个点的关联度越高；反之则越低。常用的测量的距离的方法包括：欧氏距离、切比雪夫距离和余弦距离等，KNN使用欧氏距离更为常见。例如，在二维平面中，两个点(x1,y1)和(x2,y2)的距离计算公式如下：

拓展至高维空间之后，可以得到如下距离公式：

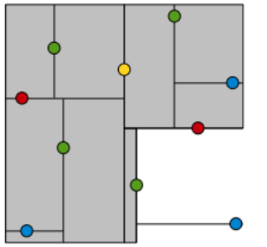
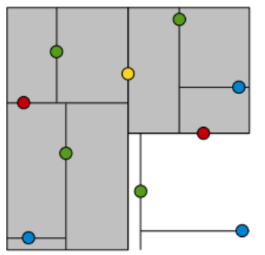
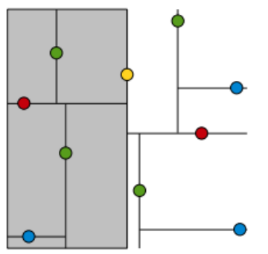
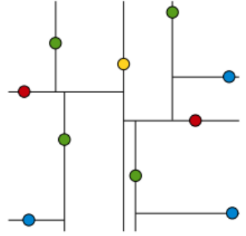
## 3.2K值的选择

K值是KNN算法中唯一一个超参数，K值的确定影响着算法的准确度。如果选择的K值过于小，那么算法在一个较小的范围内进行训练和预测，算法的近似误差较小，只有训练数据集里的与输入数据特别相似的数据才会符合预期。但是因为过于依赖最邻近的几个点的结果，那么会对预测结果产生较大误差，使KNN算法过度拟合。随着K值不断增大，那么对于较大范围内的邻近点也会对结果造成影响，所以正确率会稍微提高。但是，如果K值过于高，那么在一个较大范围内可能相近的数值类型的点会拥有不同的结果的情况，那么对于结果来说会带来更多的不确定性，所以正确率再次降低。所以，KNN算法的正确率符合一个正态分布。

## 3.3KNN算法的核心KDTree

KNN分类算法需要完成的基本任务就是，在给定的距离度量范围内，在很短的时间内找到与需要预测的实例最相近的K个邻近节点。为了更方便地完成这些任务，引入了KDTree的概念。KDTree全称K-Dimension Tree，它的特性继承与二叉树，能够存储并快速查找到K维空间中的节点。每一个维度对应划分出来的一个矩形，也对应一个KDTree的节点。利用这种二叉树的数据结构可以省去对大量节点的查找时间，提高了效率，节省了时间。

构造KDTree的步骤为：先构造根节点，使得根节点能够包含所有的后代节点的矩形区域，然后按照维度的增加，将矩形切分，生成二叉树的子节点，可以用递归方法实现。以下是KDTree查询过程示意图：



# 4. 建模优化过程及结果分析

将K值从1-20循环测试数据的正确性。最终比较出最高正确性的K值及其正确率。

# 5.最终模型性能

正确率75.5%，计算时间在1min左右。如果单算任意1K值，计算时间将在5s左右。

# 6.XAML界面

主要内容就是grid里面的datagrid控件和几个button按钮还有结果label控件显示。

# 7.函数.cs文件

难点一：

C#读取excel文件到datagrid，需要文件流读取，然后调用NPOI的接口把数据输入到datatable中，再把datatable的内容赋给datagrid。然后再把行数用lable显示出来。

难点二：、

数据清洗，把重复数据和不符合要求的数据剔除

数据归一化，把数据都转化为数值型

难点三：

测试集导入，这个方法和难点一一样，按行预测结果并和真实结果对比，并算出最终的正确率。因为KNN的K值不同正确率也不同，那就从一到二十循环一次，算出正确率最高的时候，K的值和正确率。

难点四：

KNN算法的实现，在KNN.CS文件中。主要是计算点对距离，具体内容见上面KNN的讲解。