Knn算法

传统机器学习：scikit-learn框架 sk-learn

封装了一系列的传统机器学习算法实现，而且提供一些关于特征处理的以及模型保存的功能，甚至可以外接spark做大规模数据集的机器学习

这个框架的用法基本一致，只需要改变不同的算法(类)

安装pip install scikit-learn

Knn开发思路：

1. 加载数据
2. 归一化数据
3. 计算距离 统计前k距离最短的样本中出现的类别做多的那个

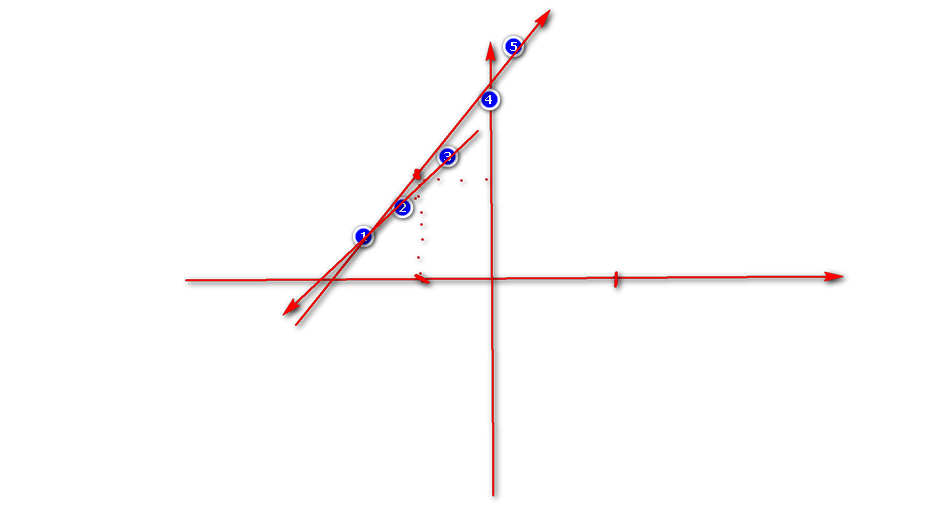
Pandas读取数据

X=pandas.read\_csv(filepath,usecols=[0,1,2],sep=**'\t'**) *# 通用方法，将数据加载为一个二维数组 usecols选择读取哪些列*Y=pandas.read\_csv(filepath,usecols=(3,),sep=**'\t'**) *# 通用方法，将数据加载为一个二维数组 usecols选择读取哪些列*

线性回归的问题：预测准确率比较低

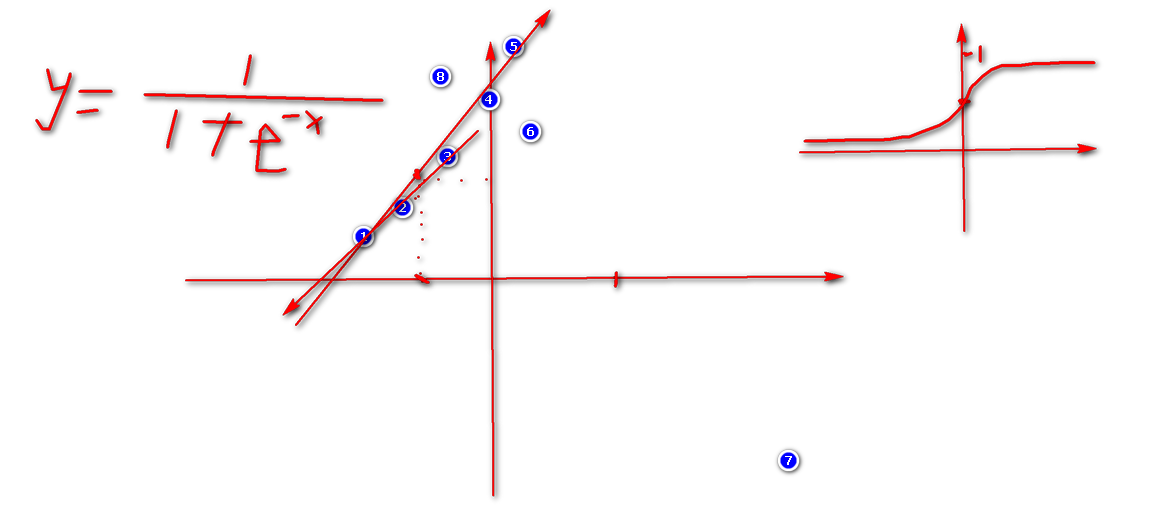
因为在现实生活中，大部分的特征与目标变量的关系都是非线性的

改进方案1：局部加权线性回归



改进方案2：逻辑回归 原生做二分类

在线性回归外套一层 sigmod函数 假设



损失函数：

F(x)=-（ylogy^+(1-y)log(1-y^)）

代价函数：

F(x)=-

使用梯度下降算法一步步更新迭代参数，求出最合适的参数

什么函数可以做损失函数？

函数的值越来越小的时候，y^越接近于y

函数的值越来越大的时候，y^越远离于y

**使用数学函数做预测的开发思路：**

1. 根据数据集提出假设（函数）
2. 找一个计算比较便利的损失函数（一个样本）
3. 计算整个样本集的成本函数
4. 使用梯度下降算法更新迭代成本函数的参数，使得成本函数的值越来越小
5. 当成本函数的损失值达到满意的程度以后，或者每次迭代损失值的变化比较小，或者达到最大迭代次数的时候，程序终止迭代，得到一组合适的参数
6. 将参数代入原来提出的假设，给定新的x的时候，根据函数算出来对应的y

**梯度下降算法：**

学习率的设定只能凭经验

当样本集很大的时候，每次更新参数都需要遍历整个样本集，训练速度较慢

改进？

随机梯度下降：

每次只用一个样本来做梯度下降，迭代较快 会有一定的精度损失

小批量（度）梯度下降：

每次用一部分样本集做梯度下降，既保证了迭代效率，又保证了精度损失

正样本和负样本

二分类的时候

0是负样本 1是正样本

有病 --》没病 提升

没病 --》有病 下降

Sk-learn 中国官方文档

http://sklearn.apachecn.org/#/docs/2