数据分析第二次作业(孙万彤)

## 线性回归

### 概念

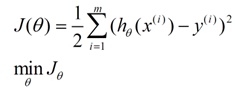
线性回归是利用数理统计中回归分析，来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法，运用十分广泛。其表达形式为y = w'x+e，e为误差服从均值为0的正态分布。线性回归假设特征和结果满足线性关系。其实线性关系的表达能力非常强大，每个特征对结果的影响强弱可以由前面的参数体现，而且每个特征变量可以首先映射到一个函数，然后再参与线性计算。这样就可以表达特征与结果之间的非线性关系。 用X1，X2..Xn 去描述feature分量，我们可以做出一个估计函数：



如果我们令X0 = 1，就可以表示为如下的向量形式：



程序也需要一个机制去评估我们θ是否比较好，需要对我们做出的h函数进行评估，这个函数就被称为损失函数（loss function）或者错误函数(error function)，用函数名为的J函数描述h函数偏离的程度

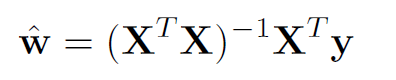


这个错误估计函数是去对x(i)的估计值与真实值y(i)差的平方和作为错误估计函数，至于为何选择平方和作为错误估计函数，从概率分布的角度讲解了该公式的来源。如何调整θ以使得J(θ)取得最小值有很多方法，其中有最小二乘法(min square)，和梯度下降法。

#### (1).最小二乘法

最小二乘法是基于均方误差最小化来进行模型求解的方法。最小二乘法就是试图找到一条直线，使的所有的样本到直线上的欧式距离之和最小。

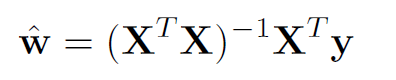
在用最小二乘法时必须要求X是满秩矩阵



#### (2).梯度下降法

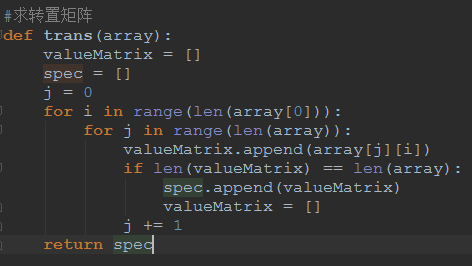
在机器学习算法中，在最小化损失函数时，可以通过梯度下降法来一步步的迭代求解，得到最小化的损失函数，和模型参数值。反过来，如果我们需要求解损失函数的最大值，这时就需要用梯度上升法来迭代了。梯度下降法和梯度上升法是可以互相转化的。比如我们需要求解损失函数f(θ)的最小值，这时我们需要用梯度下降法来迭代求解。但是实际上，我们可以反过来求解损失函数 -f(θ)的最大值，这时梯度上升法就派上用场了。

### 算法原理及其实现(用最小二乘法)

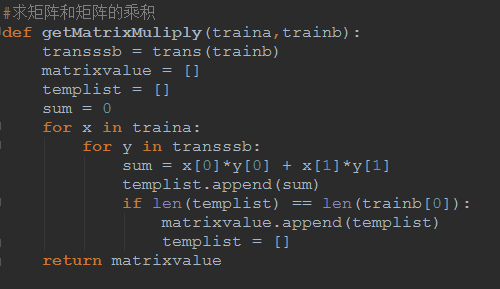


在用w求值的时候由于大部分都是矩阵操作，所以首先求矩阵的转置矩阵

求转置矩阵



求矩阵的乘积



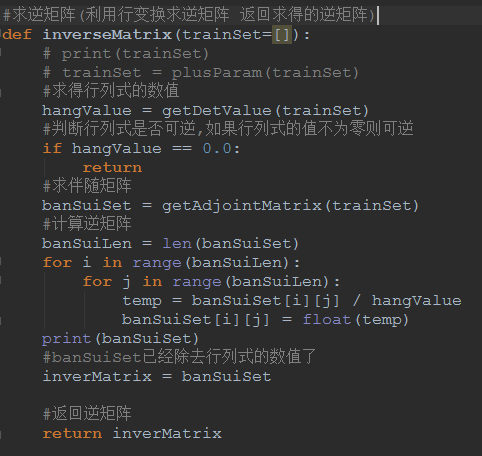
求行列式的值



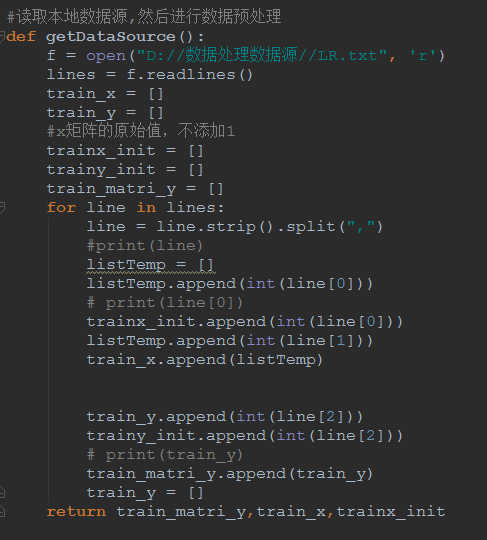
求矩阵伴随矩阵



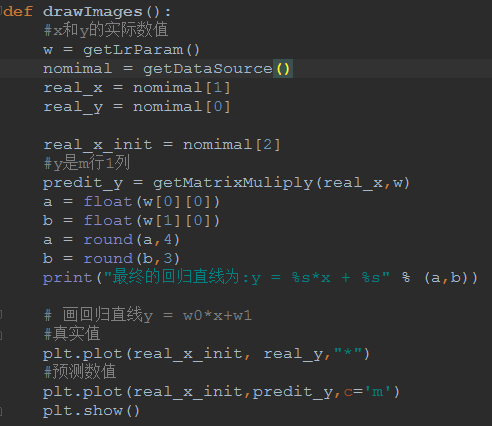
求逆矩阵



读取本地数据源进行预处理



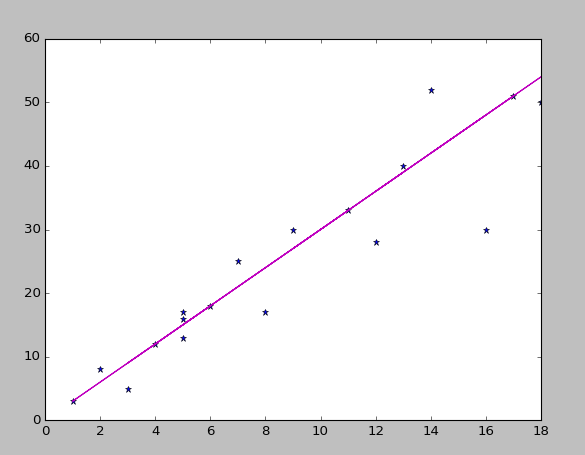
根据求得的样本特征绘制回归曲线的函数



最终求得的线性回归的方程为:



在二维平面上的表示为:



### 总结

在用最小二乘法实现线性回归算法的时候，求矩阵的逆矩阵应该是最重要的一点，在矩阵的运算中，由于要经常进行类型的转换，所以造成均方误差可能表较大，这个还有待对代码进行优化。

## 二. 决策树算法实现及结果评估程序实现

### 1.概念

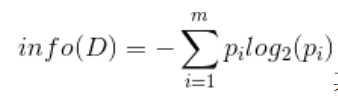
决策树（decision tree）是一个树结构（可以是二叉树或非二叉树）。其每个非叶节点  
表示一个特征属性上的测试，每个分支代表这个特征属性在某个值域上的输出，而每个叶节  
点存放一个类别。使用决策树进行决策的过程就是从根节点开始，测试待分类项中相应的特  
征属性，并按照其值选择输出分支，直至到达叶子节点，将叶子节点存放的类别作为决策结  
果。 决策树的构造就是进行属性选择度量确定各个特征属性之间的拓扑结构。 构造决策树  
的关键步骤是分裂属性。所谓分裂属性就是在某个节点处按照某一特征属性的不同划分构造  
不同的分支，其目标是让各个分裂子集尽可能地“纯” 。尽可能“纯” 就是尽量让一个分裂子集中待分类项属于同一类别。当对属性进行分类时要判断属性属于连续值还是离散值，因为决策树只能对离散属性进行处理，所以就必须对数据进行预处理，对数据进行离散化或者装箱等的操作。

构造决策树的关键内容是进行属性选择度量，属性选择度量是一种选择分裂准则，是  
将给定的类标记的训练集合的数据划分 D“最好”地分成个体类的启发式方法，它决定了拓扑结构及分裂点的选择

### 2.算法实现原理(c4.5)

#### a.选择属性的度量标准

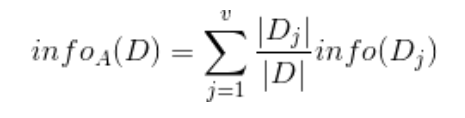
##### (1) 信息熵



其中 pi 表示第 i 个类别在整个训练数据集中出现的概率，用属于此类别元素的数量除以训练元组元素总数量作为估计。熵的实际意义表示是D 中元组的类标号信息量的均值。

##### (2) 信息增益

现在我们假设将训练元组 D 按属性 A 进行划分，则 A 对 D 划分的期望信息为：

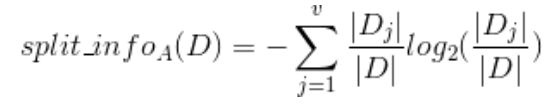


则信息增益为:

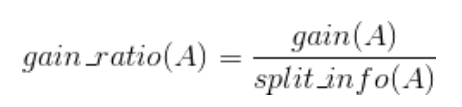


##### (3) 信息增益率

属性分裂时的信息为：



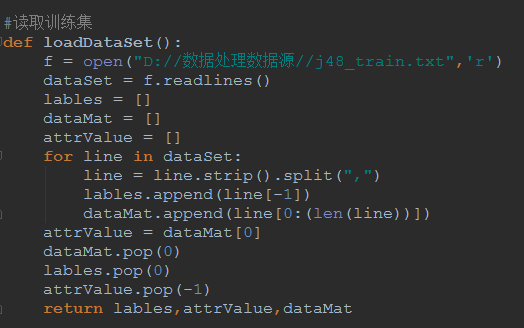
增益率:



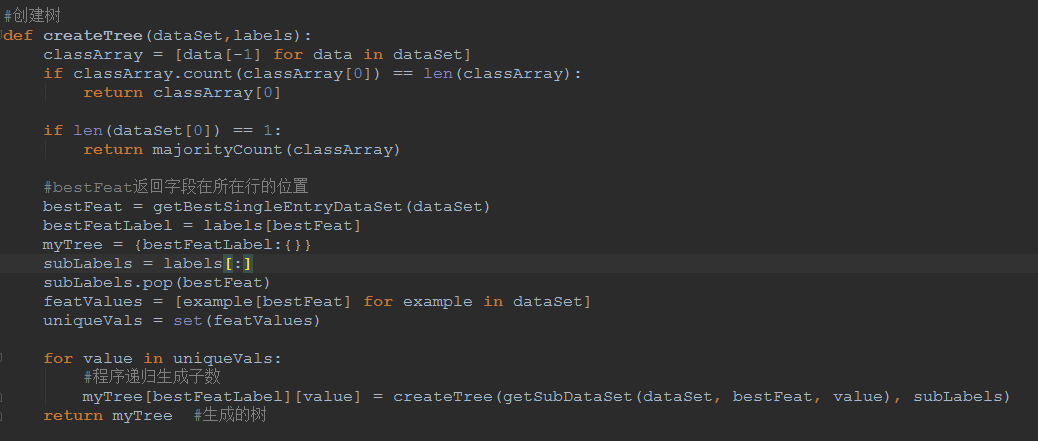
#### 

### 3.算法源码

首先读取本地训练集，进行训练集的数据预处理



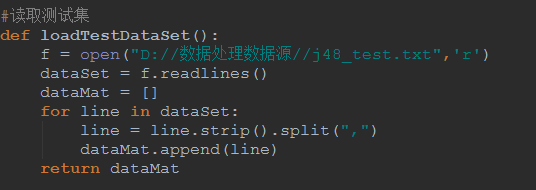
然后递归的建立决策树

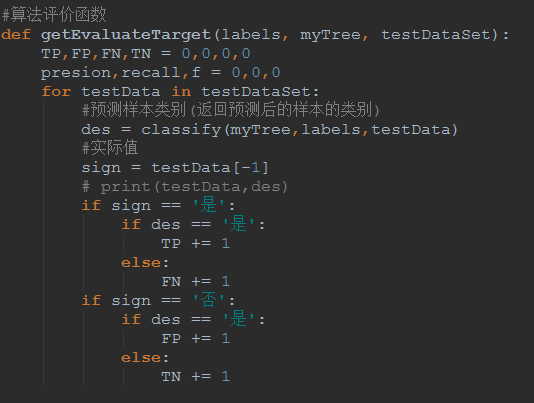


其返回值为一个dict数据结构，里边存放的是分类树的一些信息，如下图所示：

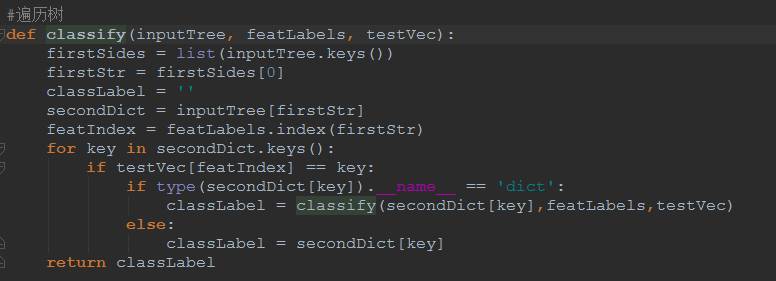


然后读取本地测试集，递归的遍历树，对测试集合进行测试

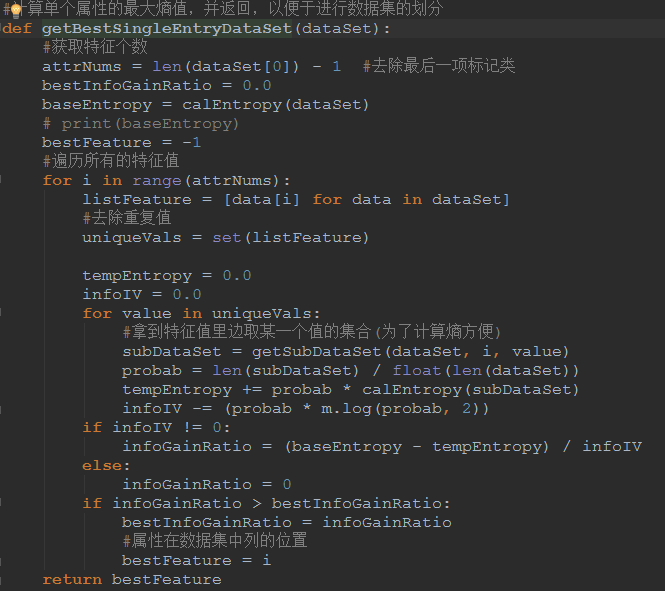




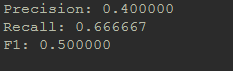
循环读取测试集，计算各项评价指标，需要遍历已经生成的树



在建立树的过程中计算各个属性熵值是比较重要的，需要递归的计算熵值，



最后得出的各项评价指标如下图所示:



### 4.总结

在进行决策树的划分中过程中，本来是需要递归的计算熵值的，然而刚开始做的时候并没有递归导致决策树少了一些分支，从而导致在对测试集进行预测的时候有些预测集并没有预测出来，导致准确率100%，后来改完之后则正常。另在测试的过程中，训练集和测试集划分是比较重要的，当用留余法的时候，如果使用单次划分，则模型的性能相对较差，需要多次随机划分数据集和训练集，最后综合每次的各种评价指标(取平均值)作为最后的评价指标。