机器学习课程实践

**第1章 概述**

1.1 课程简介

1.1.1 课程目的

1.1.2 适用对象

1.1.3 先修课程与技术

1.1.4 时间安排

1.2 课程实践任务与目标

1.2.1 任务与架构设计

1.2.2 机器学习算法集成目标

1.2.3前端与可视化设计目标

1.2.4 后端模块设计目标

1.3 本章小结

**第2章 开发环境配置**

2.1 硬件要求

2.2 数据集

2.3 软件工具

2.4 前端依赖包

2.4.1 NodeJs

2.4.2 Vue

2.5 后端依赖包

2.5.1 Python

2.5.2 Scikit-learn

2.6 Git

2.7 本章小结

**第3章 前端设计开发与部署**

3.1 HTML css 与 javascript

1. **HTML：描述页面的结构**

html的最顶层标签，就是html，需要有一个head和body，head放一些属性信息，body放页面显示的内容。html是由一些标签构成的，每个标签都有开始标签和结束标签，也有部分标签，只有开始标签，没有结束标签，标签之间可以嵌套。

1. 注释标签：<!-- 这是注释 -->
2. 标题标签：h1 h2 h3 h4 h5 h6 数字越小，字体越大越粗
3. 段落标签；<p></p> 段落标签之间会有换行，并且有明显的段落间距
4. 换行标签：<br>是一个单标签
5. 加粗：<strong></strong> <b></b>
6. 倾斜：<em></em> <i></i>
7. 删除线：<del></del> <s></s>
8. 下划线：<ins></ins> <u></u>
9. 图片标签：img 单标签

img必须要带有一个src属性，通过这个属性来指定你要显示的图片的路径，这个路径可以写绝对路径，也可以是相对路径（基准目录就是当前html所在的目录）

1. 超链接：<a href=" "></a> 通过链接可以找到另外一个资源
2. 表格标签：

table：表示整个表格

tr：表示表格的一行

td：表示一个单元格

th：表示表头单元格，会居中加粗

thead：表格的头部区域（注意要和th区分，范围是比th要大）

tbody：表格的主体区域

1. 列表标签：

有序列表 ol

无序列表 ul

列表项 li

1. 表单标签

from标签：进行前后端交互，功能是构造一个HTTP请求

input标签

文本框：<input type="text">

密码框：<input type="password">

单选框：

<input type="radio" name="gender" id="male"> <label for="male">男</label>

<input type="radio" name="gender" id="famale"> <label for="famale">女</label>

单选框之间必须具备相同的name属性，才能实现多选一效果

复选框：

<input type="checkbox" name="action">吃饭

<input type="checkbox" name="action" checked="checked">睡觉

<input type="checkbox" name="action">打游戏 -->

普通按钮：

<input type="button" value="EXO" οnclick="alert('hello,baekhyun wife')">

提交按钮：<input type="submit" value="提交按钮">

文件选择框：

<input type="file">

select下拉菜单

<select>

<option selected="selected">边伯贤</option>

<option>都暻秀</option>

<option>朴灿烈</option>

<option>吴世勋</option>

</select>

多行编辑框：<textarea cols="30" rows="10"></textarea>，自己实现滚动条

块级元素：<div> 默认是独占一行的块级元素

行内元素：<span> 默认是不独占一行的行内元素

1. **CSS：描述页面的样式**

css可以嵌入到html页面中编写，就需要有一个style  <style></style>

css具体设置的属性键值对，若干个属性都在一个{ }里，属性之间用；来分割，键和值之间使用：来分割

1. **CSS的引入样式**
2. **CSS选择器**

**描述了你要选中页面中的哪个/哪些元素，{ }的样式就是针对这些元素生效的**

**标签选择器**

**写个标签名字，标签名就表示针对当前页面中所有的指定标签，都要被选中**

**类选择器**

**可以让样式差异化效果**

**在html中使用class属性，引用对应的css类名，从而使对应的样式针对指定元素生效**

1. **复合选择器**

### **后代选择器**

元素1  元素2{ }

### **子选择器**

和后代选择器类似，只是选择了子标签

并集选择器

1. **伪类选择器**

选择元素的不同状态

：hover  鼠标放上去

：active  鼠标按下去

1. **常见属性**

字体：font-family设置字体

font-weight设置字体的粗细，值是100-900整数，数字越大字体越粗

文本颜色:

直接写单词

写成rgb/rgba形式

写成#十六进制数字

对齐:text-align

文本装饰：text-decoration给文字加上下划线（underline），上划线（overline），删除线（line-through）

文本缩进：text-indent  能够控制一段话的首行，自由缩进（2em就是两个字）

行高：line-height=文字大小+行间距

行高=顶线到顶线的距离=底线到底线的距离=中线到中线的距离=基线到基线的距离

背景：背景颜色background-col

背景图片background-image，默认是平铺的

background-repeat  控制背景图的平铺

background-position  设置背景图的位置

background-size  设置背景图片的大小

圆角矩形：border-radius

元素的显示模式：使用display属性针对行内/块级元素相互转换

盒子模型：内边距-padding

边框-border

外边距-margin

弹性布局：用来描述元素之间的关系（相对位置关系）

display：flex

justify-content  设置水平方向的排列方式

align-items  设置垂直方向的排列方式

1. **JS：描述页面的动态交互**

3.2 Nodejs 与 npm

在 Linux 上安装 Node.js 需要使用终端命令行工具。常用的安装方式有以下几种:

使用包管理器安装，例如 Ubuntu 上可以使用 apt 安装:

IMG_256sudo apt update

sudo apt install nodejs

使用 nvm (Node Version Manager) 安装，可以方便地管理多个 Node.js 版本:

IMG_257curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash

nvm install node

从 Node.js 官网下载二进制文件安装:

IMG_258wget https://nodejs.org/dist/latest/node-v14.16.2-linux-x64.tar.gz

tar -xvf node-v14.16.2-linux-x64.tar.gz

cd node-v14.16.2-linux-x64

sudo cp -R \* /usr/local/

安装后可以使用 node -v 检查 Node.js 版本。

3.3 Ts 与 VUE3

3.3.1 数据绑定

3.3.2 component

3.3.3 router

3.4 Elementui

3.5 网页设计与实现

3.6 编译与部署

3.7 本章小结

**第4章 后端设计与开发**

4.1 Scikit-learn 库的使用

4.2 机器学习与多态小练

4.2.1 实现第一个算法

4.2.2 提取公因式与构建多态

4.3 Python 魔法方法与类型

4.3.1 \_\_dict\_\_ 与 \_\_getattr\_\_

4.3.2 \_\_call\_\_

4.3.3 \_\_iter\_\_

4.3.4 Typing 与代码提示

4.4 架构设计

4.5 计算图的构建与执行

4.5.1 图的数据结构

4.5.2 图的拓扑排序

4.5.3 装饰器与注册中心

4.6 暴露后端接口

4.7 本章小结

**第5章 前后端通信**

5.1 WebSocket 协程与任务循环

5.1.1 Python 协程

5.1.2 网络与WebSocket

5.1.3 VUE WebSocket 包

5.2 日志的设计与实现

5.2.1 文本日志

5.2.2 图像日志

5.2.3 图表日志

5.3 本章小结

**第6章 机器学习十大算法设计与实现**

6.1 线性回归算法

6.1.1**基本概念**

线性回归是机器学习中有监督机器学习下的一种算法。 回归问题主要关注的是因变量(需要预测的值，可以是一个也可以是多个)和一个或多个数值型的自变量(预测变量)之间的关系。简单来说，线性回归算法以一个坐标系里一个维度为结果，其它维度为特征（如二维平面坐标系中横轴为特征，纵轴为结果），无数的训练集放在坐标系中，发现他们是围绕这一条线分布的。线性回归算法的期望，就是寻找一条直线，最大程度的拟合样本特征和样本输出标记的关系。

* 需要预测的值:即目标变量，target，y，连续值预测变量。
* 影响目标变量的因素：IMG_256，可以是连续值也可以是离散值。
* 因变量和自变量之间的关系:即模型，model，是我们要求解的。

简单线性回归属于一个算法，它所对应的公式：y=wx+b

多元线性回归公式如下：

y^=w1x1+w2x2+.....+wnxn+by^=w1x1+w2x2+.....+wnxn+b ，b是截距。

**6.1.2线性回归算法推导**

1. 误差分析

误差等于第 i 个样本实际的值减去预测的值，公式可以表达为如下：

εi=|yi−y^| 或 εi=|yi−WTxi|

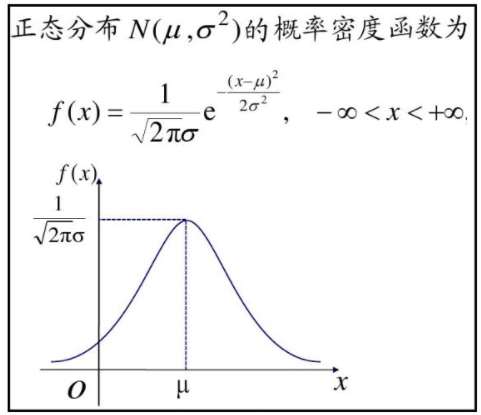
假定所有的样本的误差都是独立的，有上下的震荡，震荡认为是随机变量，足够多的随机变量叠加之后形成的分布，它服从的就是正态分布，因为它是正常状态下的分布，也就是高斯分布！均值是某一个值，方差是某一个值。 方差我们先不管，均值我们总有办法让它去等于零 0 的，因为我们这里是有截距b， 所有误差我们就可以认为是独立分布的，1<=i<=n，服从均值为 0，方差为某定值的高斯分布。机器学习中我们假设误差符合均值为0，方差为定值的正态分布.将误差定义到正太分布中.当样本足够多的时候,取样本的均值则为实际值.

1. 最大似然估计

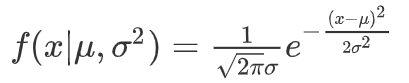
最大似然估计(maximum likelihood estimation, MLE)一种重要而普遍的求估计量的方法。最大似然估计明确地使用概率模型，其目标是寻找能够以较高概率产生观察数据的系统发生树。最大似然估计是一类完全基于统计的系统发生树重建方法的代表。

1. 高斯分布-概率密度函数

最常见的连续概率分布是正态分布，也叫高斯分布，而这正是我们所需要的，其概率密度函数如下:



正态分布公式如下：



随着参数μ和σ变化，概率分布也产生变化。 下面重要的步骤来了，我们要把一组数据误差出现的总似然，也就是一组数据之所以对应误差出现的整体可能性表达出来了，因为数据的误差我们假设服从一个高斯分布，并且通过截距项来平移整体分布的位置从而使得μ=0.

1. 误差总似然，最小二乘法MSE

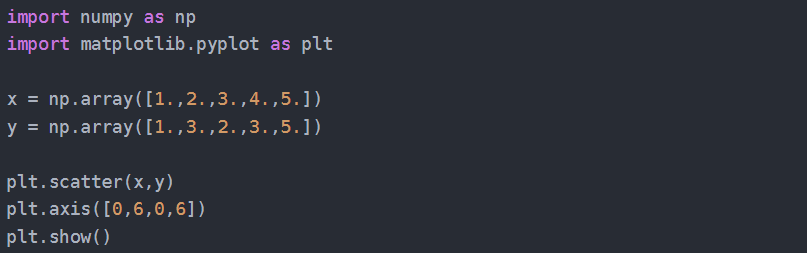
这种最小二乘法估计，其实我们就可以认为，假定了误差服从正太分布，认为样本误差的出现是随机的，独立的，使用最大似然估计思想，利用损失函数最小化 MSE 就能求出最优解！所以反过来说，如果我们的数据误差不是互相独立的，或者不是随机出现的，那么就不适合去假设为正太分布，就不能去用正太分布的概率密度函数带入到总似然的函数中，故而就不能用 MSE 作为损失函数去求解最优解了！

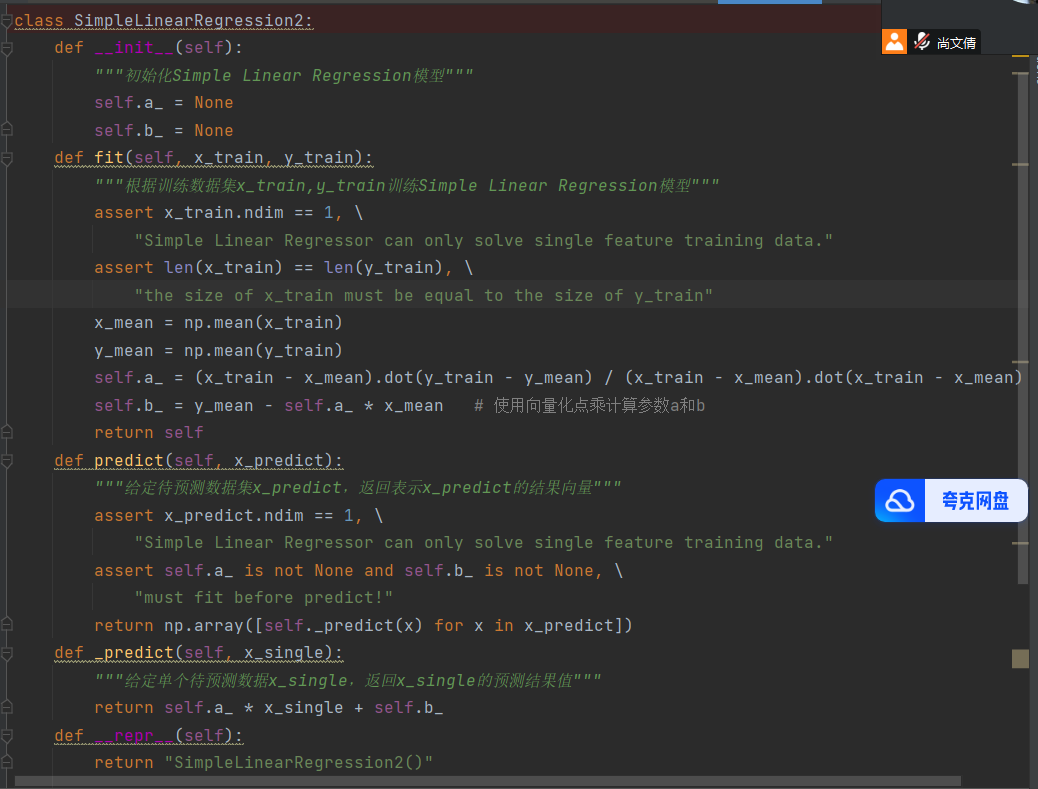
还有譬如假设误差服从泊松分布，或其他分布那就得用其他分布的概率密度函数去推导出损失函数了。

所以有时我们也可以把线性回归看成是广义线性回归。比如，逻辑回归，泊松回归都属于广义线性回归的一种，这里我们线性回归可以说是最小二乘线性回归。

**6.1.3线性回归实践**

1. 简单线性回归





调用自己的封装函数



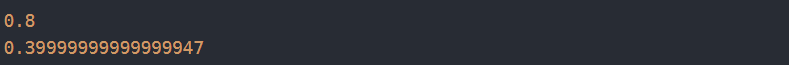
进行预测



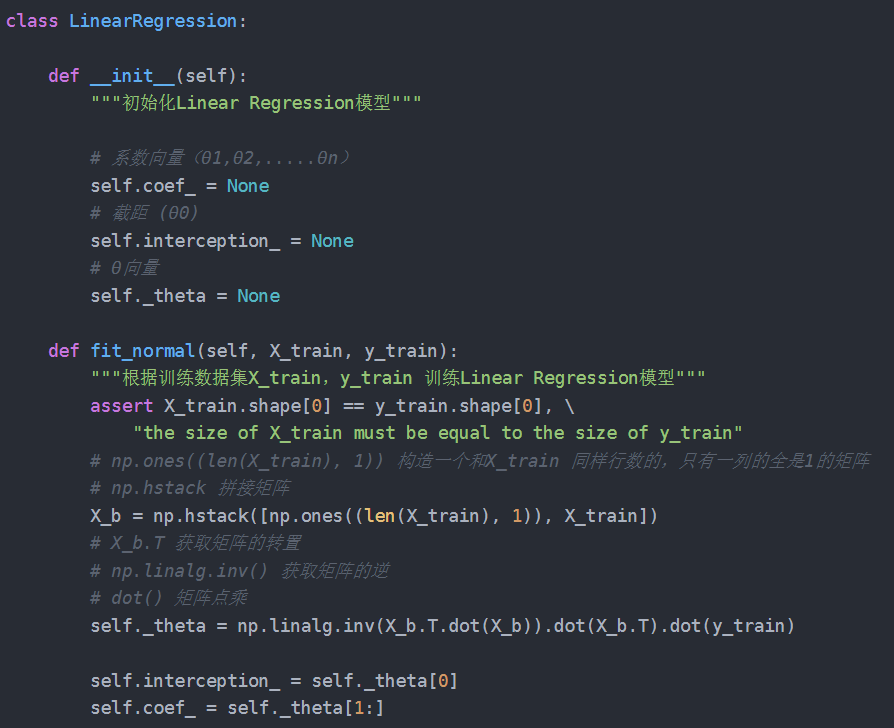


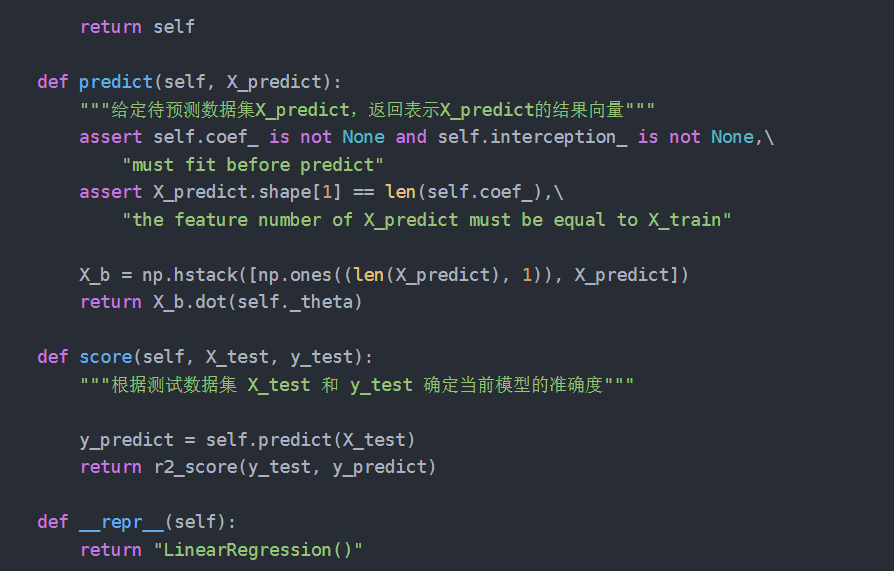
查看参数a和b



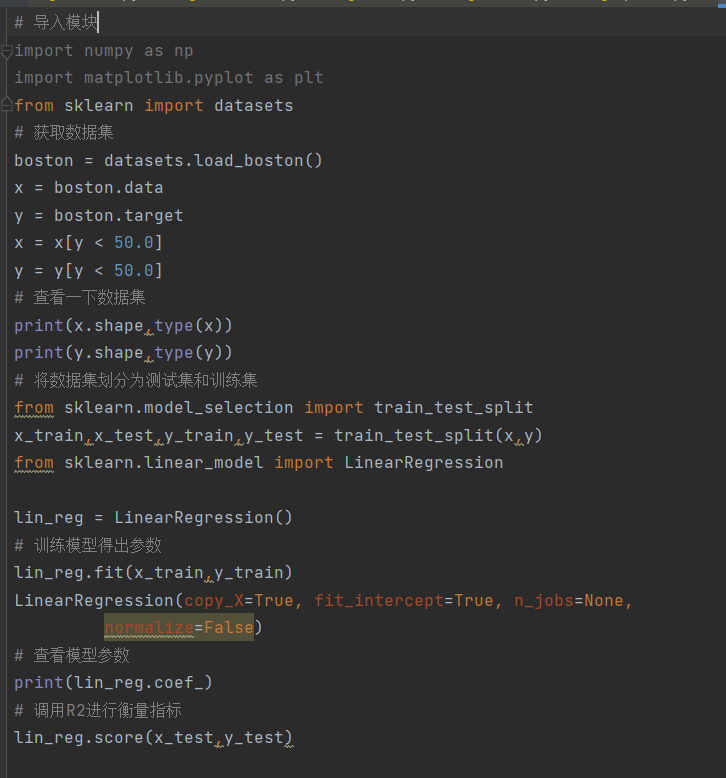


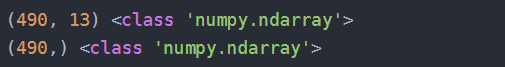
1. 多元线性回归



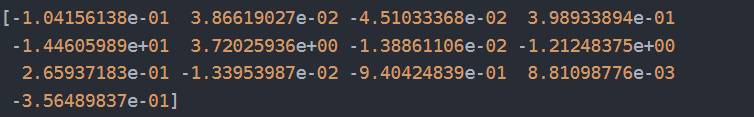


（3）波士顿房价预测





模型参数：



指标衡量：

6.2 逻辑回归算法

6.3 决策树算法

6.3.1**基本概念**

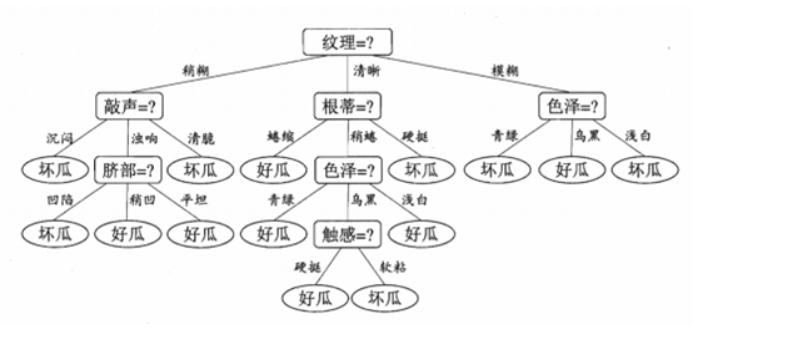
决策树算法是一种有监督学习算法，利用分类的思想，根据数据的特征构建数学模型，从而达到数据的筛选，决策的目标。

决策树( Decision Tree) 又称为判定树，是数据挖掘技术中的一种重要的分类与回归方法，它是一种以树结构（包括二叉树和多叉树）形式来表达的预测分析模型。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。

一般，一棵决策树包含一个根节点，若干个内部结点和若干个叶结点。

叶结点对应于决策结果，其他每个结点对应于一个属性测试。每个结点包含的样本集合根据属性测试的结果划分到子结点中，根结点包含样本全集，从根结点到每个叶结点的路径对应了一个判定的测试序列。决策树学习的目的是产生一棵泛化能力强，即处理未见示例强的决策树。

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，直到到达叶子节点，将叶子节点存放的类别作为决策结果。



**6.3.2决策树的构建**

1. **策略:自上而下分而治之**

自根至叶的递归过程， 在每个中间结点寻找一个“划分” 属性。

开始:构建根节点;所有训练数据都放在根节点，选择-个最优特征，按着这一特征将训练数据集分割成子集，进入子节点。

所有子集按内部节点的属性递归的进行分割。

如果这些子集已经能够被基本正确分类，那么构建叶节点，并将这些子集分到所对应的叶节点去。

每个子集都被分到叶节点.上，即都有了明确的类，这样就生成了一颗决策树。

三种停止条件:

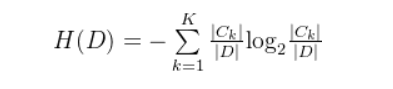
1. 当前结点包含的样本全属于同一类别，无需划分;
2. 当前属性集为空，或是所有样本在所有属性.上取值相同，无法划分;
3. 当前结点包含的样本集合为空，不能划分.

**决策树算法的核心**：如何选择最优划分属性

1. **划分选择**
2. 信息增益（ID3使用的划分方式）

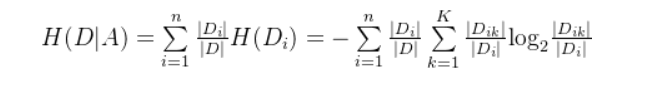
假设训练数据集D和特征A，根据如下步骤计算信息增益：

第一步：计算数据集D的经验熵：



其中，IMG_256为第类样本的数目，IMG_256为数据集D的数目。

第二步：计算特征A对数据集D的经验条件熵IMG_256：



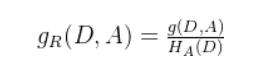
第三步：计算信息增益：

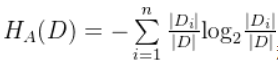


一般而言，信息增益越大，则意味着使用属性来进行划分所获得的“纯度提升” 越大。因此，我们可使用信息增益来进行决策树的划分属性选择。ID3决策树学习算法就是以信息增益为准则来选择划分属性的。

1. **信息增益率(C4.5所用的划分准则）**

特征IMG_256对于数据集IMG_257的信息增益比定义为：

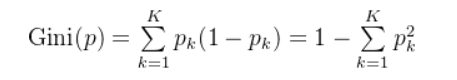


其中，称为数据集IMG_260关于IMG_261的取值熵。

增益率准则就可取值数目较少的属性有所偏好，因此，C4.5算法并不是直接选择增益率最大的候选划分属性，而是使用了一个启发式：先从候选划分属性中找出信息增益高于平均水平的属性，再从中选择增益率最高的。

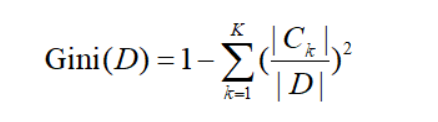
1. **基尼指数**

分类问题中，假设有个K类，样本点属于k的概率IMG_256，则概率分布的基尼指数：



二分类问题：

对给定的样本集合D，基尼指数：



CART决策树使用“基尼指数”来选择划分属性。数据集的纯度可用基尼值来度量，IMG_256越小，则数据集的纯度越高。CART生成的是二叉树，计算量相对来说不是很大，可以处理连续和离散变量，能够对缺失值进行处理。

1. **决策树剪枝**

剪枝：顾名思义就是给决策树 "去掉" 一些判断分支，同时在剩下的树结构下仍然能得到不错的结果。之所以进行剪枝，是为了防止或减少 "过拟合现象" 的发生，是决策树具有更好的泛化能力。

具体做法：去掉过于细分的叶节点，使其回退到父节点，甚至更高的节点，然后将父节点或更高的叶节点改为新的叶节点。

剪枝的两种方法：

预剪枝：在决策树构造时就进行剪枝。在决策树构造过程中，对节点进行评估，如果对其划分并不能再验证集中提高准确性，那么该节点就不要继续王下划分。这时就会把当前节点作为叶节点。

后剪枝：在生成决策树之后再剪枝。通常会从决策树的叶节点开始，逐层向上对每个节点进行评估。如果剪掉该节点，带来的验证集中准确性差别不大或有明显提升，则可以对它进行剪枝，用叶子节点来代填该节点。

预剪枝vs后剪枝

时间开销:

●预剪枝:训练时间开销降低，测试时间开销降低

●后剪枝:训练时间开销增加，测试时间开销降低

过/欠拟合风险:

●预剪枝:过拟合风险降低，欠拟合风险增加

●后剪枝:过拟合风险降低，欠拟合风险基本不变

泛化性能:

后剪枝通常优于预剪枝

1. **连续值与缺失值处理**

连续值处理：由于连续属性的可取值数目不再有限,因此不能直接根据连续属性的可取值来对结点进行划分。基本思路:连续属性离散化，常见做法:二分法。

缺失值处理：基本思路:样本赋权，权重划分

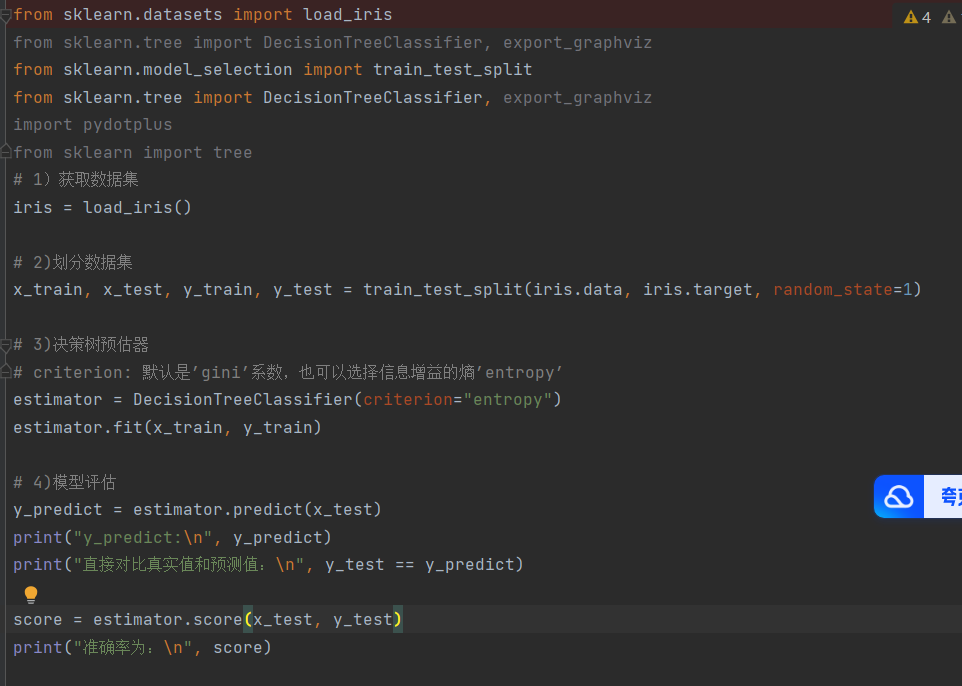
1. **决策树优缺点**

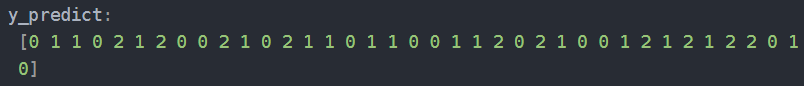
速度快:计算量相对较小，且容易转化成分类规则。只要沿着树根向下一直走到叶，沿途的分裂条件就能够唯一确定一条分类的谓词。

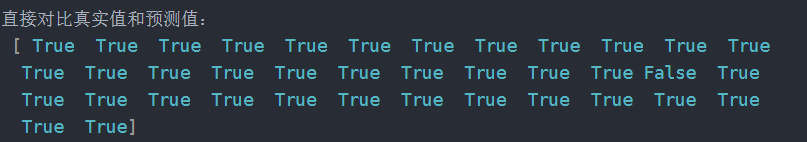
准确性高:挖掘出的分类规则准确性高，便于理解，决策树可以清晰的显示哪些字段比较重要。

非参数学习，不需要设置参数。

# **6.3.3使用决策树进行鸢尾花数据分类**







[决策树分类](https://so.csdn.net/so/search?q=%E5%86%B3%E7%AD%96%E6%A0%91%E5%88%86%E7%B1%BB&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_44525542/article/details/_blank)的准确率为：



6.4 朴素贝叶斯算法

6.5 K最近邻算法

6.6 支持向量机算法

6.7 随机森林算法

6.8 K均值聚类算法

6.9 降维算法

6.10 梯度增强算法

**第7章 团队合作与版本管理**

7.1 团队分工与接口设计

7.2 Git 版本管理

7.2.1 简介

7.2.2 基本概念

7.2.3 提交与推送

7.2.4 GitHub

7.3 合作原则与提交规范

7.4 本章小结