数据挖掘和大数据分析

Outline

① Review Decision Tree （Assignments）

② Association Rules 

③ Apriori Algorithm 

Review Decision Tree （Assignments）

作业清单（5/20）

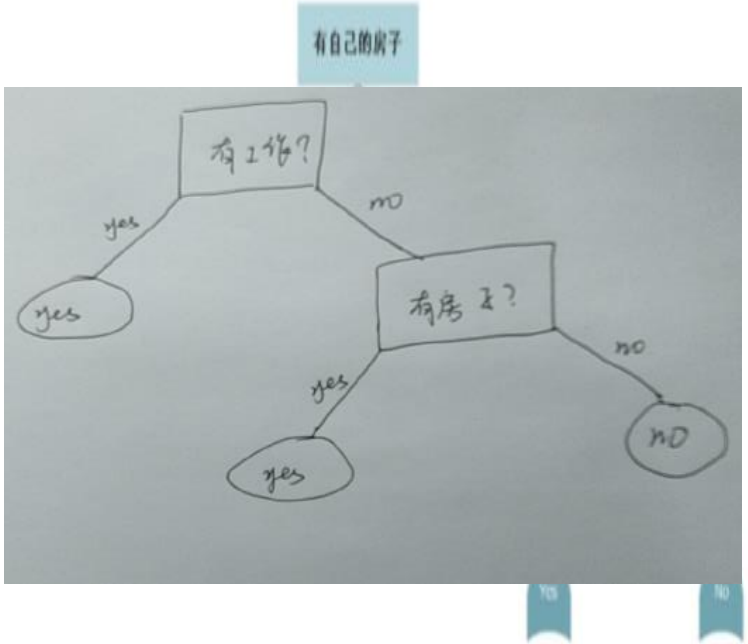
【1】 数据集如下图所示，根据我们对决策树的理解，设计一棵决策树，并输入{Age:36,Salary:H,STU:No,Credit:OK}测试数据，是否与预期结果一致？@注意，不允许直接调用Sklearn提供的决策树方法。

Age	Salary	STU	Credit	Buy Computer
<30	H	No	OK	No
<30	H	No	Good	No
30-40	H	No	OK	Yes

Review Decision Tree （Assignments）

【2】数据集如下图所示，计算这棵决策树的类别信息熵，并计算基于每个特征值的类别信息熵。根据公式计算每个特征值的信息增益，画出一棵决策树（用签字笔画出即可）。

ID	年龄	有工作	有自己的房子	信贷情况	类别(是否个给贷款)
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是



知识点：信息熵、决策树、信息增益、条件信息熵

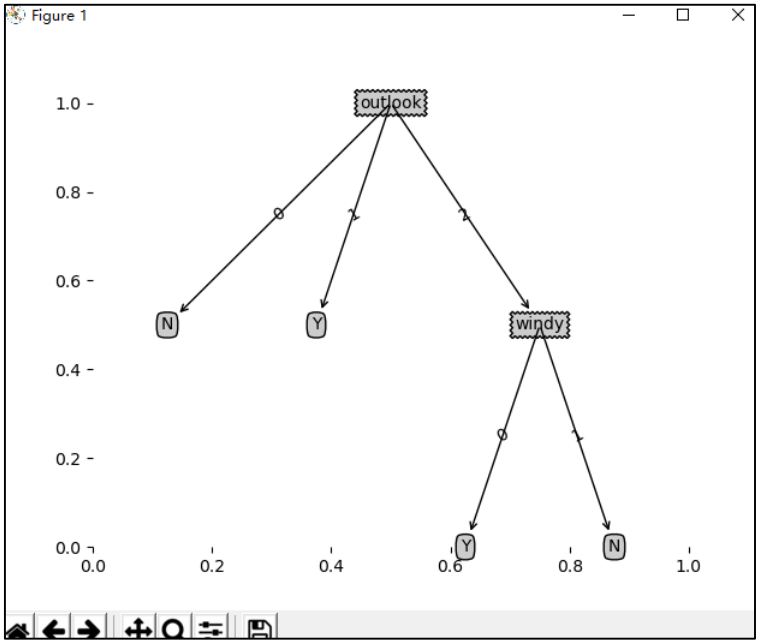
Review Decision Tree （Assignments）

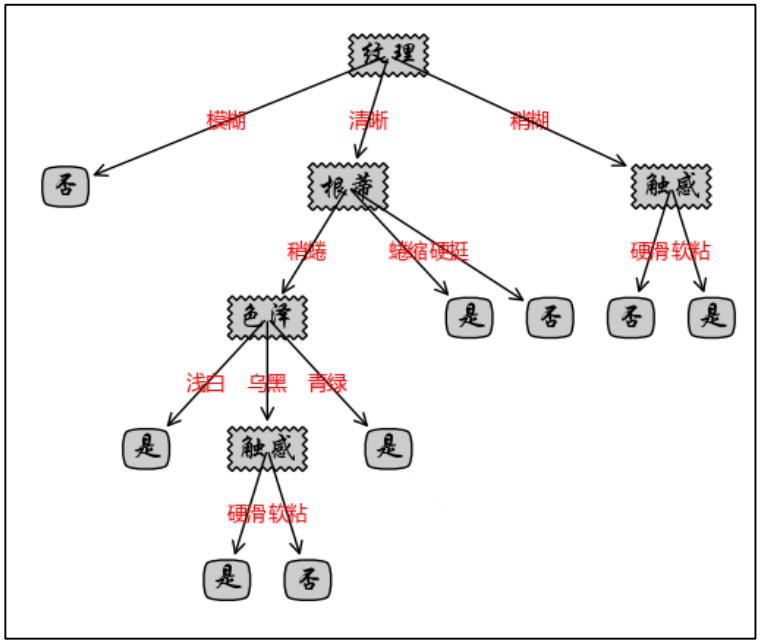
【3】 根据下述数据集，编写代码实现信息熵、条件熵、信息增益等决策树的关键环节，并撰写实验报告。（最后一列是类别：是否提供贷款）

```
dataSet = [     [0, 0, 0, 0, 'no'],                    #数据集
                [0, 0, 0, 1, 'no'],
                [0, 1, 0, 1, 'yes'],
                [0, 1, 1, 0, 'yes'],
                [0, 0, 0, 0, 'no'],
                [1, 0, 0, 0, 'no'],
                [1, 0, 0, 1, 'no'],
```

Review Decision Tree （Assignments）

【4】对问题【3】中实现的决策树，实现可视化。（选做）





Python

python™

Python语言是一种面向对象、动态数据类型的解释型语言，是数据分析师首选编程语言之一.

符号标记

x | 一个变量，其值为2. s | 字符串（string）对象.
l,L | 列表（list）对象. t | 元组（tuple）对象.
e,E | 集合（set）对象. d | 字典（dict）对象.

基本操作

x = 2 | 定义一个新的变量x，其值为2.
print | 打印输出.
| 行内注释.
""" """ | 多行注释.
? | 内省，显示对象的通用信息.
?? | 内省，显示出大部分函数的源代码.
help() | 显示一个对象的帮助文档.
%timeit | 魔术命令，计算语句的平均执行时间.

数据类型及相互转换

type(x) | 查看变量x的数据类型.
int(x) | 将变量x的数据类型转换为整型.
float(x) | 将变量x的数据类型转换为浮点型.
str(x) | 将变量x的数据类型转换为字符串.
bool(x) | 将变量x的数据类型转换为布尔型.
isinstance(x, float) | 检测变量x是否为浮点型，返回一个布尔型数值.

算术运算符

x + 5 | 加，计算结果为7.
x - 5 | 减，计算结果为-3.
x * 5 | 乘，计算结果为10.
x / 5 | 除，Python 2.x版本的计算结果为0，Python 3.x版本的计算结果为0.4.
x ** 2 | 幂运算，即 x^2，计算结果为4.

布尔型

False None 0 " " () [] {} | False值.
and | 等价于“&”，表示“且”.
or | 等价于“|”，表示“或”.
not | 表示“非”.

字符串

s = u" " | 定义Unicode字符串.
s = r" " | 定义原始字符串，避免字符串中的字符转义，在正则表达式中经常使用到.
s = "cookdata" | 定义值为“cookdata”的变量s.
len(s) | 返回s的字符个数8.
s.lower() | 将字符串s中的字母全部转换为小写.
s.upper() | 将字符串s中的字母全部转换为大写.
s.capitalize() | 将字符串s中的首个字符转换为大写，其余部分转换为小写.
s.replace('k', 'l') | 使用字符“l”替换掉s中所有的字符“k”，返回结果为cooldata.
s.strip() | 去除掉s最前面和最后面的空格.
s.split("\t") | 使用制表符“\t”分割字符串s.
'%s is No.%d' %(s, 1) | 取出s的值和数值1依次放入字符串%s is No.%d的相应位置，返回结果为cookdata is No.1.
'{} is No.{}'.format(s, 1) | 取出s的值和数值1依次放入{}相应位置，返回结果为cookdata is No.1.

列表

l = ['c', 'o', 'o', 'k', 1] | 创建一个包含字符元素c、o、o、k和整数1的列表.
list() | 创建空列表，或将其他数据结构转换为列表.
l[0] | 返回列表的第一个元素，即字符c.
l[-1] | 返回列表的最后一个元素，即1.
l[1:3] | 列表切片，返回包含原列表的第二个元素和第三个元素的列表['o' , 'o'].
len(l) | 返回列表的元素个数.
l[::-1] | 将列表进行逆序排列.
l.reverse() | 将列表进行逆序排列.
l.insert(1, 'b') | 在指定的索引位置插入‘b’.
l.append() | 在列表末尾添加元素.
l.extend(L) | 等价于“l+L”，将列表L中的元素依次添加到l的末尾.
l.remove() | 删除列表中的某个元素.
l.pop() | 等价于“del l[]”，删除列表中对应索引位置的元素.
" ".join(['c', 'o', 'o', 'k']) | 将列表中的各个字符串元素用空格连接起来并转换为字符串，返回结果为c o o k.

Python

数据嗨客

文件读写

读取文件

f = open(filename,mode) | 返回一个文件对象f，读文件“model = r”，写文件“model = w”.
f.read(size) | 返回包含前size个字符的字符串.
f.readline() | 每次读取一行，返回该行字符串.
f.readlines() | 返回包含整个文件内容的列表，列表的元素为文件的每一行内容所构成的字符串.
f.close() | 关闭文件并释放它所占用的系统资源.

```
with open("cookdata.txt", "r") as f:
    content = f.readlines()
```

在with主体块语句执行完后，自动关闭文件并释放占用的系统资源.

```
import csv
f = open("cookdata.csv", "r")
csvreader = csv.reader(f)
content_list = list(csvreader)
```

读取csv文件，并把数据存储为一个嵌套列表（列表的元素仍是一个列表）content_list.

写入文件

```
f.write(s)
print(s, file=f)
```

两种等价的方式，将字符串s写入文件对象f中.

函数

```
def sum(a, b=1):
    return a+b
```

定义求和函数sum()，该函数要求输入位置参数a，带默认值的参数b为可选参数，其默认值为1，函数返回结果为a+b的计算结果.

sum(1, b=10) | 执行sum()函数，返回结果为11.
def sum(*args, **kwargs) | 不定长参数，*args接收包含多个位置参数的元组，**kwargs接收包含多个关键字参数的字典.
obj.methodname | 一个方法是一个“属于”对象并被命名为obj.methodname的函数.

map() 和 lambda

map(func, sequence) | 将函数依次作用在序列的每个元素上，把结果作为一个新的序列返回.
lambda a, b:a+b | 匿名函数，正常函数定义的语法糖，a和b为输入参数，a+b为函数主体和返回的值.

模块

import module as alias | 导入模块，并取一个别名，使用alias.func即可调用模块内的函数.
from module import * | 导入模块中的所有函数，直接使用函数名即可调用模块内函数.
from module import func1, func2 | 导入模块中的部分函数，使用func1可直接调用该函数.

面向对象的类

```
class Athlete(object):
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def capitalize_name(self):
        return self.name.capitalize()
```

使用关键字class定义Athlete类，该类继承object类，初始化变量为self.name和self.age，并定义类的capitalize_name方法将self.name的首字母变成大写.

a = Athlete('james', '23') | 创建实例a.
a.name | 返回a的name属性，返回‘james’.
a.age | 返回a的age属性，返回23.
a.capitalize_name() | 调用a的实例方法capitalize_name，返回‘James’.
isinstance(a, Athlete) | 判断a是否是Athlete类的实例.

编码和解码

ASCII | 基于拉丁字母的一套电脑编码系统，不包含中文、日文等非英语字符.

GBK | GBK兼容ASCII，同时收录中文、日文等，使用两个字节编码一个汉字.

Unicode | 收录超过十万个字符，统一了所有语言文字的标准编码集，包括UTF-8和UTF-16两种实现方式.

s = u'中文' | 定义Unicode字符串‘中文’.
s.encode('utf-8') | 使用UTF-8编码集将Unicode字符串s编码为str字符串.

```
s = '中文'
s.decode('utf-8')
```

定义str字符串‘中文’，并解码为Unicode字符串.

```
import chardet
chardet.detect(s)
```

检测字符串的编码方式.

异常处理

语法错误 | Syntax Errors，代码编译时检测到的错误.

异常 | Exceptions，代码运行时检测到的错误，如类型错误（TypeError）、数值错误(ValueError)、索引错误（IndexError）和属性错误（AttributeError）等.

```
try:
    statement
except:
    pass
```

先尝试运行try部分的statement语句，如果能够正常运行，则跳过except部分，如果运行出现错误，则跳过错误代码运行except部分的pass语句，“放过”错误.

```
try:
    statement
except Exception, e:
    print "Error Happened: %s" %e
```

指定要处理的运行时错误类型，如果指定的错误出现，则打印获错信息，e是发生错误时的具体错误信息.

```
try:
    statement1
except (Exception1, Exception2), e:
    statement2 # 发生指定错误时的处理
else:
    statement3 # 正确时运行
finally:
    statement4 # 无论对错都运行
```

try…except…else…finally句式.

抛出异常

raise Exception('Oops!') | 主动抛出异常Exception，错误提示信息为‘Oops!’.
assert statement, e | 若statement语句运行结果为True，则继续运行代码，否则抛出e的错误提示信息.

正则表达式

正则表达式模块 re

```
import re
raw_s = r'\d{17}[\d|x]|\d{15}'
pattern = re.compile(raw_s)
re.search(pattern, s)
```

用于匹配身份证号。
首先使用原始字符串定义正则表达式模式；然后编译原始字符串为正则表达式Pattern对象；最后对整个字符串s进行模式搜索，如果模式匹配，则返回MatchObject的实例，如果该字符串没有模式匹配，则返回None.

re.search(r'\d{17}[\d|x]|\d{15}', s) | 将Pattern编译过程与搜索过程合而为一.
re.match(pattern, s) | 从字符串s的起始位置匹配一个模式，如果起始位置匹配不成功，则返回None.
re.findall(pattern, s) | 返回一个包含所有满足匹配模式的子串的列表.
re.sub(pattern, repl, s) | 使用替换字符串repl替换匹配到的子字符串.
re.split(pattern, s) | 利用满足匹配模式的子串将字符串s分隔开，并返回一个列表.

日期处理

```
from datetime import datetime
format = "%Y-%m-%d %H:%M:%S"
date_s = datetime.strptime(s,format)
```

将日期字符串按照指定日期格式转换为datetime类.

date_s.year | 获取日期字符串中的年份.
date_s.month | 获取日期字符串中的月份.
datetime.now() | 获取现在的日期和时间.

NumPy

数据嗨客

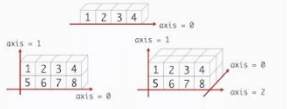
NumPy

NumPy包是Python科学计算中的核心工具包之一，它提供了高性能多维数组结构 - ndarray和用来操控这些数组的各种工具和函数.

导入包（NumPy 1.14.0）

```
import numpy as np
```

NumPy中的维度方向用axis表示.



创建数组

`np.zeros((2))` | 2个0的一维数组.
`np.zeros((2, 3))` | 6个0的二维数组.
`np.ones((2, 3, 4))` | 24个1的三维数组.
`np.empty((3))` 创建未初始化的一维数组
`np.eye(3)` | 创建3 × 3的单位矩阵.
`np.array([[1,2], [3,4]])` | 二维数组.
`np.array([1,0], dtype=np.bool)` | 一维布尔数组.
`np.array([[[1,2], [3,4]], [[5,6], [7,8]]])` | 三维数组.
`np.full((2,3), 7)` | 数值全部为7的2乘3二维数组.
`np.arange(4)` | 从0到3的一维数组.
`np.arange(5, 25, 5)` | 从5（包含）到25（不包含），步幅为5的一维数组.
`np.linspace(0, 9, 10)` | 从0（包含）到9（包含），总长为10的一维数组.
`np.random.random((2, 3))` | 从区间[0.0, 1.0)中随机抽取6个数，填入2乘3的数组中.
`np.cos(np.linspace(0, 2*np.pi, 10))` 包含十个0到2π余弦函数值的一维数组.

数组的结构

`arr.shape` | 数组的维度.
`len(arr)` | 数组的长度.
`arr.ndim` | 数组维度的大小.
`arr.size` | 数组中元素的个数.
`arr.dtype` | 数组数据的类型.
`arr.dtype.name` | 数组数据类型的名称.
`arr.astype('float')` | 将数据类型转换为其它类型.

数据类型

`np.int64` | 64位整数型.
`np.float32` | 标准双精度浮点型.
`np.complex` | 由128位浮点代表的复数.
`np.bool` | 布尔型.
`np.object` | Python中object类型.
`np.nan` | NA值，为浮点型
`np.string_` | 定长字符串类型.
`np.unicode_` | 定长unicode类型.

文件读取

`np.save('my_arr', arr)` | 以NumPy格式将数组保存到本地.
`np.savetxt('my_arr.txt', arr)` | 以文本格式将数组到本地.
`np.load('my_arr.npy')` | 读取本地NumPy数组文件.
`np.loadtxt('my_arr.txt')` | 读取本地文本数组文件.
`np.genfromtxt('file.csv', delimiter=',')` | 将其它文件文件转换成NumPy数组.

数组切片

说明：arr = np.arange(0,100).reshape(20,5)

`arr[4]` | 提取第5个（行）的元素.
`arr[1,0]` | 提取第2行第1列的元素.
`arr[[0,2], [1,3]]` | 提取位于第一行第二列[0,1]和第三行第四列[2,3]的两个元素.
`arr[1][0]` | 提取第2行第1列的元素.
`arr[0:5]` | 提取第1行到第5行元素.
`arr[:5]` | 提取前5行的元素.
`arr[0:10:2]` | 提取第1，3，5，7，9行元素.
`arr[::-1]` | 反序数组.
`arr[0:2, 2]` | 提取前2行的第3列的元素.
`arr[:,:]` | 提取所有行和所有列.
`arr[4, ...]` | 提取第5行的元素.
`arr[arr<50]` | 提取所有小于50的元素.

数组的操作

说明：arr = np.array([[1,2],[3,4]])

`arr.T` | 数组转置，[[1, 3], [2, 4]].
`arr.ravel()` | 扁平化数组，[1, 2, 3, 4].
`arr.reshape(4, 1)` | 重塑数组，元素个数需保持一致，[[1],[2],[3],[4]].
`arr.resize((2, 3))` | 重塑原数组，元素个数可不同，[[1, 2, 3],[4, 0, 0]].
`np.append(arr, [[5,6]])` | 扁平化后插入新数组[[5,6]]，若指定axis参数，则会按指定方向插入，[1, 2, 3, 4, 5, 6].
`np.insert(arr, 3, [5,6])` | 扁平化后按索引插入新数组[[5,6]]，若指定axis参数，则会按指定方插入，[1, 2, 3, 5, 6, 4].
`np.delete(arr, 1)` | 扁平化后按索引删除元素，若指定axis参数则会按维度方删除，[1, 3, 4].
`np.concatenate((arr, arr), axis=1)` 按指定维度方向（需已存在）合并多个数组，[[1, 2, 1, 2],[3, 4, 3, 4]].
`np.hstack((arr, arr))` | 按水平方向合并多个数组.
`np.vstack((arr, arr))` | 按竖直方向合并多个数组.
`np.split(arr,2)` | 按第0维度方向将数组切成2份，[[1, 2]]和[[3, 4]].
`np.split(arr,2,1)` | 按第1维度方向将数组切成2份，[[1],[3]]和[[2],[4]].
`arr.tolist()` | 将数组转换成列表.

数组的运算

`np.nan is np.nan` | 返回True.
`np.nan == np.nan` | 返回False.
`a == b` | 逐元素比较，返回等维度布尔数组.
`a < 2` | 逐元素比较，返回等维度布尔数组.
`a+b, a-b, a*b, a/b` | 逐元素进行加、减、乘、除运算.
`a.dot(b)` | 数量积运算.
`a.max(a), a.min(a), a.mean(a), np.std(a)` | 求数组整体的最大、最小、均值和标准差.
`a.max(axis=0), a.min(axis=1)` | 按指定维度方向求最大值、最小值.
`a.sort(axis=0)` | 按指定维度方向增序排序数组.
`a.copy()` | 深拷贝数组.
`a.view()` | 创建数组的view.
`np.copy(a)` | 浅拷贝数组.
`np.array_equal(a, a)` | 数组间整体比较，返回单个布尔值.
`np.exp(a), np.sqrt(a), np.sin(a), np.log(a)` | 逐元素地做幂、开平方、正弦、自然对数运算.

在NumPy 1.14版本中不支持的运算包括：
a.median()和a.corrcoef().

Pandas

数据嗨客

pandas

基于NumPy构建，利用它的高级数据结构和操作工具，可使数据分析工作变得更加便捷高效。

符号标记

s | 一个Series对象.
df | 一个DataFrame对象.

导入包（Pandas 0.20.1）

```
import numpy as np
import pandas as pd
```

基本数据结构

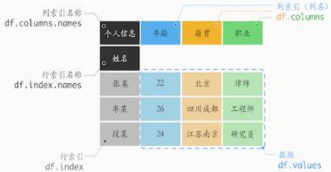
Series

一维数据结构，包含行索引和数据两个部分.


```
s = pd.Series([14, 15, 17],
        index=[u'张某', u'李某', u'段某'])
```

DataFrame

二维数据结构，包含带索引的多列数据，各列的数据类型可能不同.




```
df = pd.DataFrame([[22, u'北京', u'律师' ],
                   [26, u'四川成都', u'工程师'],
                   [24, u'江苏南京', u'研究员']],
                  index=[u'张某', u'李某', u'段某'],
                  columns=[u'年龄', u'籍贯', u'职业'])
```

文件读写

从文件中读取数据（DataFrame）

pd.read_csv() | 从CSV文件读取.

参数: file, 文件路径; sep, 分隔符; index_col, 用作行索引的一列或者多列; usecols, 选择列; encoding, 字符编码类型, 通常为'utf-8'.

pd.read_table() | 从制表符分隔文件读取，如TSV.
pd.read_excel() | 从Excel文件读取.
pd.read_sql() | 从SQL表或数据库读取.
pd.read_json() | 从JSON格式的URL或文件读取.
pd.read_clipboard() | 从剪切板读取.

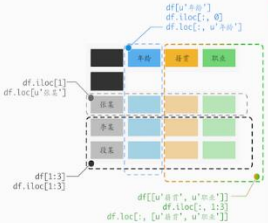
将DataFrame写入文件

df.to_csv() | 写入CSV文件.

参数: file, 文件路径; sep, 分隔符; columns, 写入文件的列; header, 是否写入表头; index, 是否写入索引.

df.to_excel() | 写入Excel文件.
df.to_sql() | 写入SQL表或数据库.
df.to_json() | 写入JSON格式的文件.
df.to_clipboard() | 写入剪切板.

数据索引

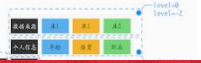


df[5:10] | 通过切片方式选取多行.
df[col_label] or df.col_label | 选取列.
df.loc[row_label, col_label] | 通过标签选取行/列.
df.iloc[row_loc, col_loc] | 通过位置（自然数）选取行/列.

数据探索

s.unique() | 唯一值.
s.value_counts() | 唯一值及其计数.
df.head(n) | 前n行数据.
df.tail(n) | 后n行数据.
df.samp
df.shap
df.info
df.desc

层次化索引



df[u'即
业"是否
df[df[u
df.loc[
表达式筛
df.filt
df.drop
df.drop
df.drop
df.sele
clude)

inplace=
axis=0,
df.asty
df.inde
df.colu
df.set_
df.rese
df.rein
据index
df.rena
索引或列

df.sort
df.sort
进行升序
df.sort
据"A"列
df.rank

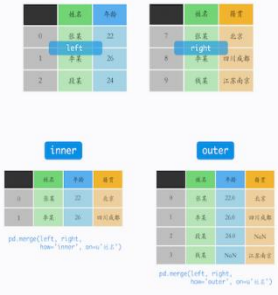
pd.merge(left, right) | 类数据库的数据融合操作.

参数: how, 融合方式, 包括左连接、右连接、内连接（取以）和外连接; on, 连接键; left_on, 左键; right_on, 右键; left_index, 是否将left行索引作为左键; right_index, 是否将right行索引作为右键.

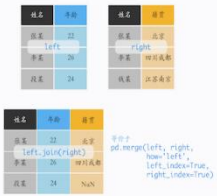
Pandas
数据嗨客

数据合并

数据融合（merge）



数据融合（join）

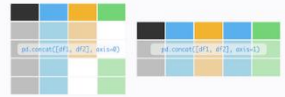


left.join(right) | 在索引上的数据融合操作.

数据融合（combine_first）

left.combine_first(right)
在数据融合的同时（行索引和列索引取并集），使用right中的值填补left中相应位置的缺失值.

轴向连接



pd.concat([df1, df2]) | 轴向连接多个DataFrame.

轴向旋转和数据转换

轴向旋转

df.stack() | 将数据的列"旋转"为行.
df.unstack() | 将数据的行"旋转"为列.

数据转换

s.map() | 利用函数或映射进行数据转换.
df.applymap() | 利用函数或映射进行数据转换.
df.replace() | 替换元素值.
df.columns.map(str.upper) | 将列名转换为大写.

数据聚合与分组运算

df.groupby('A') | 根据"A"列的值进行分组，并返回一个GroupBy对象.
df.groupby(['A', 'B']) | 根据"A"列和"B"列的值组合进行分组，并返回一个GroupBy对象.
df.groupby('A')['B'].mean() | 根据"A"列的值进行分组，并计算每一组"B"的均值.
df.groupby('A').agg(np.mean) | 根据"A"列的值进行分组，并计算每一组中各列的均值.
df.apply(np.mean) | 对DataFrame的每一列求均值.
df.apply(np.mean, axis=1) | 对DataFrame的每一行求均值.
df.pivot_table(index='A', values=['B', 'C'], aggfunc=mean) | 创建数据透视表，根据"A"列的值进行分组，并计算每一组中"B"和"C"的均值.

离散化和哑变量编码

```
pd.cut(s, bins=[0, 5, 10],
       labels=['A', 'B'])
```
离散化和面元划分.

```
pd.qcut(s, 2,
        labels=['A', 'B'])
```
等频离散化.

```
pd.get_dummies(df, columns=['A'],
               drop_first=True)
```
对"A"列进行哑变量编码，并去掉第一个编码特征.

串联方法

通过调用多个Pandas方法将多个处理过程串联起来.

```
pd.merge(left, right, how='outer',
         indicator=True)
  .query('_merge == "left_only"')
  .drop(['_merge'], axis=1)
```
保留在left中但不在right中的行.

文本数据规整

s.str.method
矢量化字符串方法，可跳过缺失值，遇缺失值将不报错.

s.str.lower() | 将所有元素转换为全部小写.
s.str.upper() | 将所有元素转换为全部大写.
s.str.replace() | 替换值.
s.str.split(" ").str[0] | 空格分割，并取列表中的第一个元素.
s.str.split(" ").str.get(0) | 空格分割，并取列表中的第一个元素.
s.str.split(" ", expand=True) | 空格分割，并扩展为多列.
s.str.contains(r'\d') | 利用正则表达式判断是否元素包含数字.
s.str.extract(r'(\d.)', expand=False) | 利用正则表达式提取每个元素中的数字.

数据可视化

df.plot(kind='hist') | 设置kind参数绘制直方图.
df.plot.line() | 调用对象接口绘制折线图.
df.plot.scatter('x', 'y') | 绘制x-y散点图.

包括图形: 面积图df.plot.area; 条形柱状图df.plot.bar; 横向柱状图df.plot.barh; 盒图df.plot.box; 和核密度图df.plot.kde; 蜂巢图df.plot.hexbin; 饼图df.plot.pie.

Scikit-learn

数据嗨客

Scikit-learn（0.19.1）是基于NumPy、SciPy和Matplotlib的开源Python机器学习包，它封装了一系列数据预处理、机器学习算法、模型选择等工具，是数据分析师首选的机器学习工具包。

符号标记

X_train | 训练数据.	y_train | 训练集标签.
X_test | 测试数据.	y_test | 测试集标签.
X | 完整数据.	y | 数据标签.

基本建模流程

```
① 导入工具包
from sklearn import datasets, preprocessing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
② 加载数据
boston = datasets.load_boston()
X = boston.data
y = boston.target
③ 训练集-测试集划分
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
④ 数据预处理
scaler = preprocessing.StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
⑤ 模型构建与拟合
lr = LinearRegression()
lr.fit(X_train, y_train)
⑥ 模型预测与评价
y_pred = lr.predict(X_test)
r2_score(y_test, y_pred)
```

加载数据

✓ Scikit-learn支持以NumPy的arrays对象、Pandas对象、SciPy的稀疏矩阵及其他可转换为数值型arrays的数据结构作为其输入，前提是数据必须是数值型的.

✓ sklearn.datasets模块提供了一系列加载和获取著名数据集如siris尾花、波士顿房价、Olivetti人脸、MNIST数据集等的工具，也包括了一些toy data和S型数据等的生成工具.

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
```

训练集-测试集划分

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
                                   X, y, random_state=12,
                                   stratify=y, test_size=0.3)
```

将完整数据集的70%作为训练集，30%作为测试集，并使得测试集和训练集中各类别数据的比例与原始数据集比例一致（stratify分层策略），另外可通过设置 shuffle=True 提前打乱数据.

数据预处理

标准化

```
from sklearn.preprocessing import StandardScaler
① 构建转换器实例
scaler = StandardScaler()
② 拟合及转换
scaler.fit_transform(X_train)
```

部分数据预处理方法	对应的sklearn的类
最小最大标准化	MinMaxScaler
One-Hot编码	OneHotEncoder
归一化	Normalizer
二值化（单个特征转换）	Binarizer
标签编码	LabelEncoder
缺失值填补	Imputer
多项式特征生成	PolynomialFeatures

特征选择

```
from sklearn import feature_selection as fs
```

fs.SelectKBest(score_func, k) | 过滤式（Filter），保留得分排名前k的特征（top k方式）.

fs.RFECV(estimator, scoring="r2") | 封装式（Wrapper），结合交叉验证的递归特征消除法，自动选择最优特征个数.

fs.SelectFromModel(estimator) | 嵌入式（Embedded），从模型中自动选择特征，任何具有coef_或者feature_importances_的基模型都可以作为estimator参数传入.

有监督学习算法

回归

```
from sklearn.linear_model import LinearRegression
① 构建模型实例
lr = LinearRegression(normalize=True)
② 训练模型
lr.fit(X_train, y_train)
③ 作出预测
y_pred = lr.predict(X_test)
```

部分回归算法	对应的sklearn的类
LASSO	linear_model.Lasso
Ridge	linear_model.Ridge
ElasticNet	linear_model.ElasticNet
回归树	tree.DecisionTreeRegressor

分类

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(max_depth=5)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_prob = clf.predict_proba(X_test)
```

决策树分类算法示例，对于二分类问题，y_prob为每个样本预测为“0”和“1”类的概率.

部分分类算法	对应的sklearn的类
逻辑回归	linear_model.LogisticRegression
支持向量机	svm.SVC
朴素贝叶斯	naive_bayes.GaussianNB
K近邻	neighbors.NearestNeighbors

集成

sklearn.ensemble模块包含了一系列基于集成思想的分类、回归和离群值检测方法.

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=20)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
y_prob = clf.predict_proba(X_test)
```

部分集成算法	对应的sklearn的类
AdaBoost	ensemble.AdaBoostClassifier
	ensemble.AdaBoostRegressor
基于梯度提升	ensemble.GradientBoostingClassifier
	ensemble.GradientBoostingRegressor

无监督学习算法

聚类

sklearn.cluster模块包含了一系列无监督聚类算法.

```
from sklearn.cluster import KMeans
① 构建聚类实例
kmeans = KMeans(n_clusters=3, random_state=0)
② 拟合
kmeans.fit(X_train)
③ 预测
kmeans.predict(X_test)
```

部分聚类算法	对应的sklearn的类
DBSCAN	cluster.DBSCAN
层次聚类	cluster.AgglomerativeClustering
谱聚类	cluster.SpectralClustering

降维

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit_transform(X_train)
```

通过主成分分析（PCA）将原始数据映射到2维空间.

部分降维方算法	对应的sklearn的类
线性判别分析	discriminant_analysis
	LinearDiscriminantAnalysis
核主成分分析	decomposition.KernelPCA
局部线性映射	manifold.LocallyLinearEmbedding
t-SNE	manifold.TSNE

模型评价

sklearn.metrics模块包含了一系列用于评价模型的评分函数、损失函数以及成对数据的距离度量函数.

```
from sklearn.metrics import accuracy_score
metrics.accuracy_score(y_true, y_pred)
```

对于测试集而言，y_test即是y_true，大部分函数都必须包含真实值y_true和预测值y_pred，基于排放的关系，以下不再重复写这两个参数.

回归模型评价

metrics.mean_absolute_error() | 平均绝对误差MAE.

metrics.mean_squared_error() | 均方误差MSE.

metrics.r2_score() | 决定系数R^2.

分类模型评价

metrics.accuracy_score() | 正确率.

metrics.precision_score() | 各类精确率.

metrics.f1_score() | F1值.

metrics.log_loss() | 对数损失或交叉熵损失.

metrics.confusion_matrix | 混淆矩阵.

metrics.classification_report | 含多种评价的分类报告.

交叉验证及超参数调优

交叉验证

```
from sklearn.model_selection import cross_val_score
clf = DecisionTreeClassifier(max_depth=5)
scores = cross_val_score(clf, X_train, y_train,
                         cv=5, scoring='f1_weighted')
```

使用5折交叉验证对决策树模型进行评估，使用的评分函数为F1值.

✓ sklearn提供了部分带交叉验证功能的模型类如LassoCV、LogisticRegressionCV等，这些类包含cv参数.

超参数调优——网格搜索

```
from sklearn.model_selection import GridSearchCV
svc = svm.SVC()
params = {'kernel':['linear', 'rbf'], 'C':[1, 10]}
grid_search = GridSearchCV(svc, params, cv=5)
grid_search.fit(X_train, y_train)
grid_search.best_params_
```

在参数网格上进行穷举搜索，方法简单但是搜索速度慢（超参数较多时），且不容易找到参数空间中的局部最优.

超参数调优——随机搜索

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
svc = svm.SVC()
param_dist = {'kernel':['linear', 'rbf'], 'C':randint(1, 20)}
random_search = RandomizedSearchCV(svc, param_dist, n_iter=10)
random_search.fit(X_train, y_train)
random_search.best_params_
```

在参数空间中进行随机搜索，选取空间中的100个点进行建模（可从scipy.stats常见分布如正态分布norm、均匀分布uniform中随机采样得到），时间耗费较少，更容易找到局部最优.

Matplotlib

数据嗨客

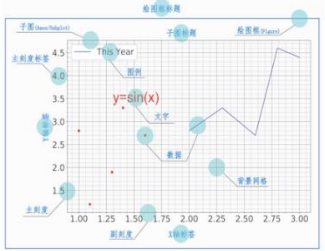
matplotlib

Matplotlib是Python中一个非常有名的绘图包，它能快速绘制具有印刷品质的图形，并且支持跨平台运行和交互式操作.

导入包（matplotlib 2.1.1）

```
import matplotlib.pyplot as plt
```

图形的各元素名称如下：



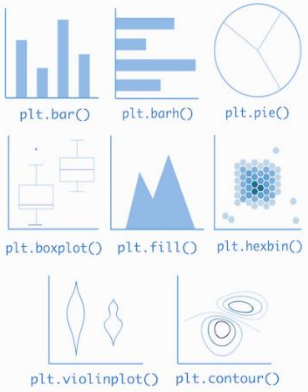
基本概念

绘图框 是图形的最高容器，所有图形必须放置在绘图框中.

子图 是绘图框中所包含的图形，即便绘图框只包含一幅图，也称之为子图.

元素 是组成子图的部件，从子图最内部的数据线条到外围的坐标轴标签等都属于元素.

图形样式



图形设置

小工具

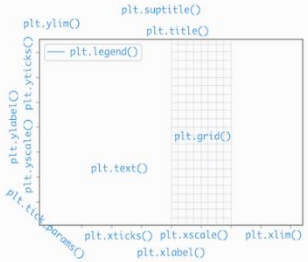
```
plt.show()  | 显示图形.
plt.savefig('fig.eps') | 保存图形.
plt.figure(figsize=(12, 8)) | 创建12×8的绘图框.
```

刻度调整

```
plt.xlim(0,1)|设置x轴显示范围，y轴同理.
plt.tick_params(axis='x', size=50, labelsize=20) | 修改主刻度及其标签样式.
plt.grid(which='major')|添加背景网格.
plt.xscale('log')|设置轴比例，y轴同理.
plt.xticks(np.arange(0,1,0.2))|修改主刻度标签范围，y轴同理.
```

文字调整

```
plt.title('title') | 添加子图标题.
plt.legend(['ln','pt']) | 添加图例.
plt.xlabel('x') | 添加x轴标签，y轴同理.
plt.text(1,1,'t')|在(1,1)位置添加文字.
plt.annotate('t', xy=(1,2), xytext=(3,4), arrowprops=dict()) | 在子图中添加带有指向箭头的文字注释.
plt.suptitle('T') | 添加绘图框标题.
```




```
from matplotlib.font_manager import FontProperties
myfont=FontProperties(fname='FangSong.TTF')
plt.title(u'简单的正弦图形', fontproperties=myfont) | 使用中文字体
```

高级绘图

绘制子图

```
plt.subplot(1,2,1) | 创建1×2的子图矩阵，当前绘制第一幅子图.
plt.tight_layout() | 自动调整子图间距.
plt.subplots_adjust() | 调整子图大小.
```

高级操作

```
fig = plt.figure() | 创建绘图框并返回绘图框对象.
ax = fig.add_subplot(121) | 对创建好的绘图框对象添加1×2的子图矩阵并返回第一个子图对象.
fig, ax = plt.subplots(1,2)|创建1×2的子图矩阵并同时返回绘图框和两个子图对象，其效果等于前两者之和.
ax.plot(), ax.bar(), ... | 向子图中添加图形，其余样式同理.
ax.set_title(), ax.set_xlabel(),... 对子图添加标题、标签等，其余元素同理.
plt.gca() | 返回当前绘图框中的子图对象.
plt.gcf() | 返回当前绘图框对象.
ax.invert_xaxis() | 反序x轴，y轴同理.
ax.xaxis.tick_top() | 将x轴主刻度标签移动到上方，y轴同理.
ax.tick_params() | 设置坐标轴刻度.
ax.set_frame_on(False) | 关闭子图边框.
ax.set_axis_off() | 关闭子图所有坐标轴.
```

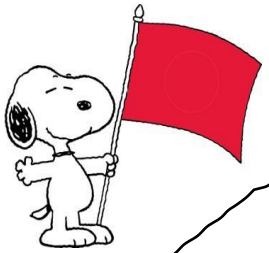
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = Axes3D(fig) | 创建3D子图对象.
```

常用颜色

	tomato		coral
	whitesmoke		lightcoral
	silver		mistyrose
	rosybrown		seashell
	firebrick		peachpuff
	darksalmon		ivory
	sienna		olive
	sandybrown		lawngreen
	bisque		wheat
	tan		skyblue
	moccasin		violet
	floralwhite		indigo
	gold		slateblue
	palegreen		lightcyan
	darkgreen		pink
	seagreen		aqua
	cyan		navy
	darkcyan		lavender
	cornsilk		springgreen

Review

How do you understand DM、AI、ML?



Aim: DM

AI

Machine Learning

Learning Aim

Learning Method

Review

Data Mining Methods

- **分类**
  - 分类问题属于预测性的问题，它跟普通预测问题的区别在于其预测的结果是类别。
  - 典型三种算法：Logistic回归；决策树；神经网络
  - 应用场所：判定类别等
- **聚类**
  - 指把一组个体按照相似性归成若干类别。
  - 典型算法：基于欧氏距离；K-means算法
  - 应用场所：市场细分
- **关联**
  - 挖掘发现大量数据中项集之间有趣的关联或相关联系
  - 典型算法：Apriori算法；FP-Growth算法；
  - 应用场所：交叉销售（购物篮-啤酒与尿布）

DATA ANALYTICS:

DATA MINING AND BIG DATA

—— Mining Rules 1

Association Rules

New Concepts

Frequent Item_sets 频繁项集

Support 支持度

Confidence 置信度

Support (X => Y) = |{X，Y}| / |Items| Confidence (X => Y) = |{X，Y}| / |X_Items|

单选题 1分

设置

In 10 days, there were 4 days when Xiao Ming ate a hamburger and bought a watermelon at the same time. What is support？（ ）

A 40%

B 20%

C 50%

D 60%

提交

单选题 1分

设置

In 10 days, Xiao Ming ate hamburgers for 8 days. During the 8 days, Xiao Ming also bought watermelon for 4 days. What is the confidence? ( )

A 40%

B 20%

C 50%

D 60%

提交

Association Rules

Vanilla Wafer => Banana, Raw cream butter

[Support = 1%, Confidence = 40%]

Banana => Vanilla Wafer, Raw cream butter

[Support = 1%, Confidence = 10%]

Support : Find Frequent Item_sets

Confidence : Find Association rules

Association Rules

篮子	商品1	商品2	商品3
1	香草威化	香蕉	狗粮
2	香蕉	面包	酸奶
3	香蕉	苹果	酸奶
4	香草威化	香蕉	生奶油
5	面包	香草威化	酸奶
6	牛奶	面包	香蕉
7	香草威化	苹果	香蕉
8	酸奶	苹果	香草威化
9	香草威化	香蕉	牛奶
10	香蕉	面包	花生酱

$$Lift\ (A => B) = \frac{Support(A \Rightarrow B)}{Support(A) * Support(B)}$$

Confidence (Vanilla Wafer => Banana) = 4/6 = 67%

Confidence (Banana => Vanilla Wafer) = 4/8 = 50%

Apriori算法

First, Set a Support initial threshold.

Set 1- Dataitems list.

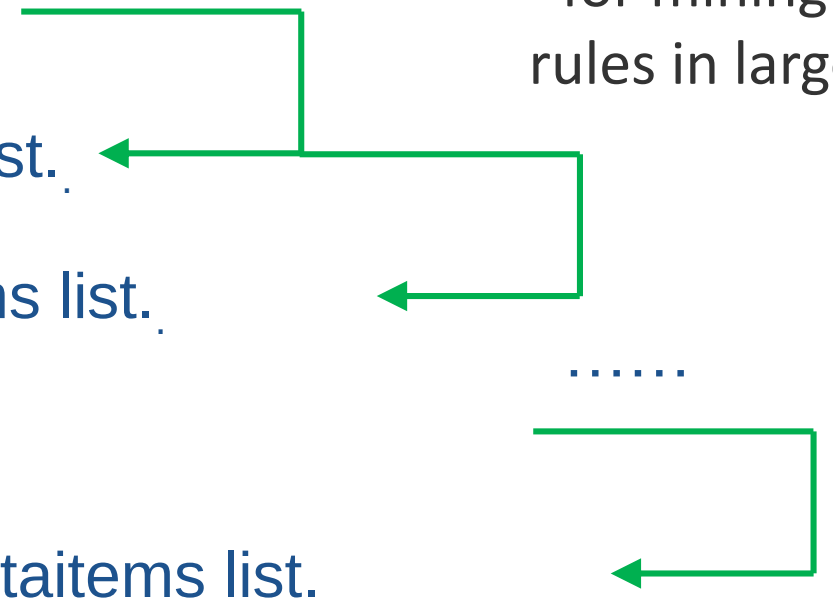
Set 2- Dataitems list.

Set 3- Dataitems list.

……

Set n- Dataitems list.

1994 —“Fast algorithms for mining association rules in large databases”



Objectives (课程目标)

O1：Master Machine Learning classic algorithm

O2：Master classic Statics Methods

O3：Data Mining Methods

O4：Solve some Business examples

Tools

Basics

Application

W1: Statics

W2: Statics Quiz1

W3: Machine Learning

W4: Machine Learning

W5: Machine Learning Quiz2

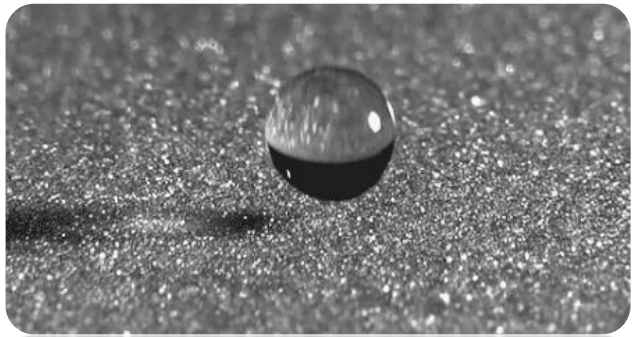
W6: Data Ming Rules

W7: Spark + Dig Data Quiz 3

W8: Example Final Exam

30%

70%



贵在坚持！