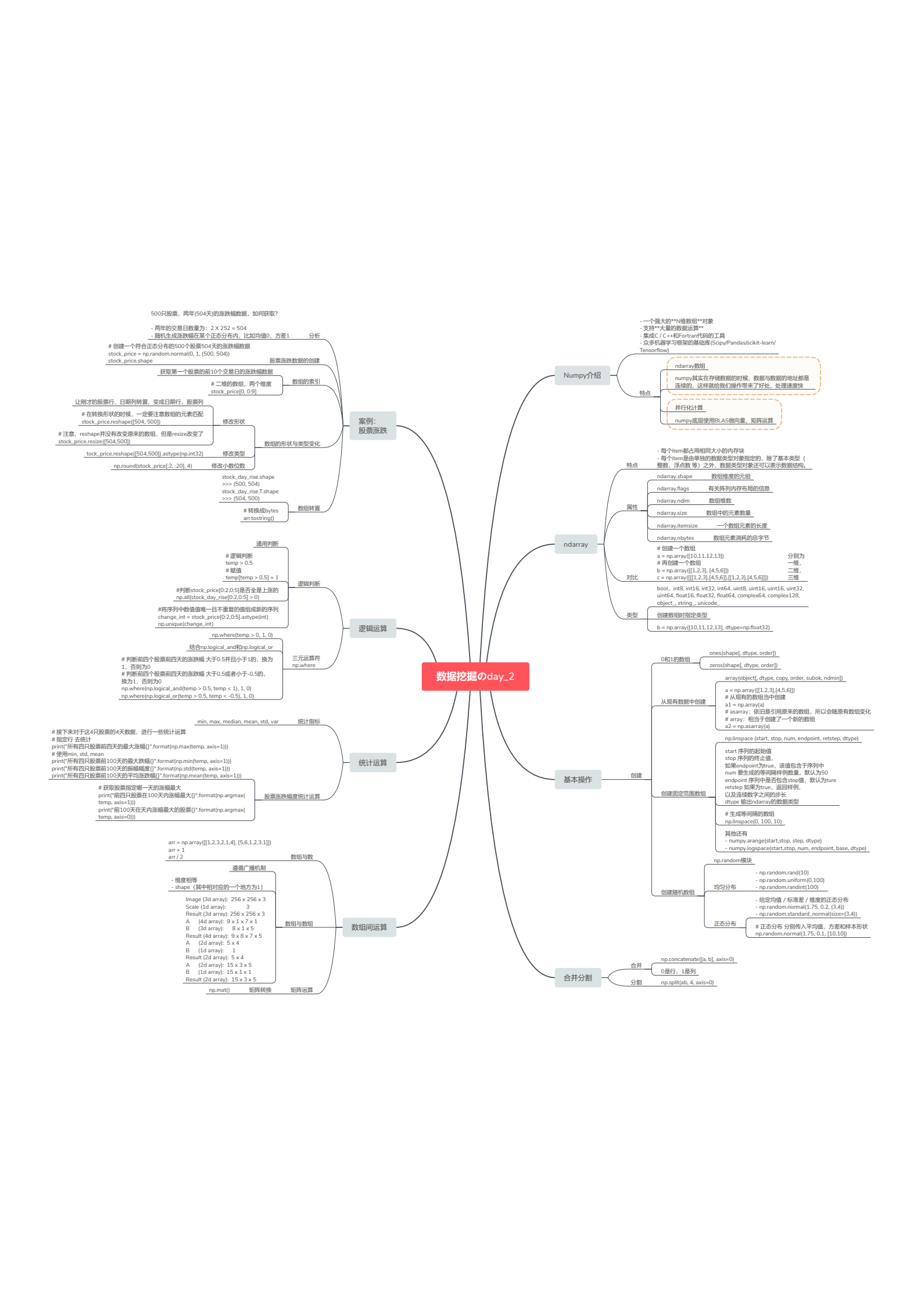
数据挖掘のday_2
Numpy介绍
- 一个强大的**N维数组**对象
- 支持**大量的数据运算**
- 集成C / C++和Fortran代码的工具
- 众多机器学习框架的基础库(Scipy/Pandas/scikit-learn/Tensorflow)
特点
ndarray数组
numpy其实在存储数据的时候，数据与数据的地址都是连续的，这样就给我们操作带来了好处，处理速度快
并行化计算
numpy底层使用BLAS做向量，矩阵运算
ndarray
特点
- 每个item都占用相同大小的内存块
- 每个item是由单独的数据类型对象指定的，除了基本类型（整数，浮点数 等）之外，数据类型对象还可以表示数据结构。
属性
ndarray.shape 数组维度的元组
ndarray.flags 有关阵列内存布局的信息
ndarray.ndim 数组维数
ndarray.size 数组中的元素数量
ndarray.itemsize 一个数组元素的长度
ndarray.nbytes 数组元素消耗的总字节
对比
# 创建一个数组
a = np.array([10,11,12,13]) 分别为一维，
# 再创建一个数组
b = np.array([[1,2,3], [4,5,6]]) 二维，
c = np.array([[[1,2,3],[4,5,6]],[[1,2,3],[4,5,6]]]) 三维
类型
bool，int8, int16, int32, int64, uint8, uint16, uint16, uint32, uint64, float16, float32, float64, complex64, complex128, object_, string_, unicode_
创建数组时指定类型
b = np.array([10,11,12,13], dtype=np.float32)
基本操作
创建
0和1的数组
ones(shape[, dtype, order])
zeros(shape[, dtype, order])
从现有数据中创建
array(object[, dtype, copy, order, subok, ndmin])
a = np.array([[1,2,3],[4,5,6]])
# 从现有的数组当中创建
a1 = np.array(a)
# asarray：依旧是引用原来的数组，所以会随原有数组变化
# array：相当于创建了一个新的数组
a2 = np.asarray(a)
创建固定范围数组
np.linspace (start, stop, num, endpoint, retstep, dtype)
start 序列的起始值
stop 序列的终止值，
如果endpoint为true，该值包含于序列中
num 要生成的等间隔样例数量，默认为50
endpoint 序列中是否包含stop值，默认为ture
retstep 如果为true，返回样例，
以及连续数字之间的步长
dtype 输出ndarray的数据类型
# 生成等间隔的数组
np.linspace(0, 100, 10)
其他还有
- numpy.arange(start,stop, step, dtype)
- numpy.logspace(start,stop, num, endpoint, base, dtype)
创建随机数组
np.random模块
均匀分布
- np.random.rand(10)
- np.random.uniform(0,100)
- np.random.randint(100)
正态分布
- 给定均值 / 标准差 / 维度的正态分布
- np.random.normal(1.75, 0.2, (3,4))
- np.random.standard_normal(size=(3,4))
# 正态分布 分别传入平均值，方差和样本形状
np.random.normal(1.75, 0.1, (10,10))
合并分割
合并
np.concatenate([a, b], axis=0)
0是行，1是列
分割
np.split(ab, 4, axis=0)
案例：股票涨跌
500只股票，两年(504天)的涨跌幅数据，如何获取?
分析
- 两年的交易日数量为：2 X 252 = 504
- 随机生成涨跌幅在某个正态分布内，比如均值0，方差1
股票涨跌数据的创建
# 创建一个符合正态分布的500个股票504天的涨跌幅数据
stock_price = np.random.normal(0, 1, (500, 504))
stock_price.shape
数组的索引
获取第一个股票的前10个交易日的涨跌幅数据
二维的数组，两个维度
stock_price[0, 0:9]
数组的形状与类型变化
修改形状
让刚才的股票行、日期列转置，变成日期行，股票列
# 在转换形状的时候，一定要注意数组的元素匹配
stock_price.reshape([504, 500])
# 注意，reshape并没有改变原来的数组，但是resize改变了
stock_price.resize([504,500])
修改类型
tock_price.reshape([504,500]).astype(np.int32)
修改小数位数
np.round(stock_price[:2, :20], 4)
数组转置
stock_day_rise.shape
>>> (500, 504)
stock_day_rise.T.shape
>>> (504, 500)
# 转换成bytes
arr.tostring()
逻辑运算
逻辑判断
通用判断
逻辑判断
temp > 0.5
赋值
temp[temp > 0.5] = 1
#判断stock_price[0:2,0:5]是否全是上涨的
np.all(stock_day_rise[0:2,0:5] > 0)
#将序列中数值值唯一且不重复的值组成新的序列
change_int = stock_price[0:2,0:5].astype(int)
np.unique(change_int)
三元运算符 np.where
np.where(temp > 0, 1, 0)
结合np.logical_and和np.logical_or
# 判断前四个股票前四天的涨跌幅 大于0.5并且小于1的，换为1，否则为0
# 判断前四个股票前四天的涨跌幅 大于0.5或者小于-0.5的，换为1，否则为0
np.where(np.logical_and(temp > 0.5, temp < 1), 1, 0)
np.where(np.logical_or(temp > 0.5, temp < -0.5), 1, 0)
统计运算
统计指标
min, max, median, mean, std, var
# 接下来对于这4只股票的4天数据，进行一些统计运算
# 指定行 去统计
print("所有四只股票前四天的最大涨幅{}".format(np.max(temp, axis=1)))
# 使用min, std, mean
print("所有四只股票前100天的最大跌幅{}".format(np.min(temp, axis=1)))
print("所有四只股票前100天的振幅幅度{}".format(np.std(temp, axis=1)))
print("所有四只股票前100天的平均涨跌幅{}".format(np.mean(temp, axis=1)))
股票涨跌幅度统计运算
# 获取股票指定哪一天的涨幅最大
print("前四只股票在100天内涨幅最大{}".format(np.argmax(temp, axis=1)))
print("前100天在天内涨幅最大的股票{}".format(np.argmax(temp, axis=0)))
数组间运算
数组与数
arr = np.array([[1,2,3,2,1,4], [5,6,1,2,3,1]])
arr + 1
arr / 2
数组与数组
遵循广播机制
- 维度相等
- shape（其中相对应的一个地方为1）
Image (3d array): 256 x 256 x 3
Scale (1d array): 3
Result (3d array): 256 x 256 x 3
A (4d array): 9 x 1 x 7 x 1
B (3d array): 8 x 1 x 5
Result (4d array): 9 x 8 x 7 x 5
A (2d array): 5 x 4
B (1d array): 1
Result (2d array): 5 x 4
A (2d array): 15 x 3 x 5
B (1d array): 15 x 1 x 1
Result (2d array): 15 x 3 x 5
矩阵运算
矩阵转换
np.mat()