## 《程序设计与人工智能》 模块与函数参考

## 第一部分：numpy以及数组常用函数

### 1.创建数组

**numpy.array( object, dtype=None, ndmin=0 )**

**object**：可被转换成数组的其它数据对象。比如列表。

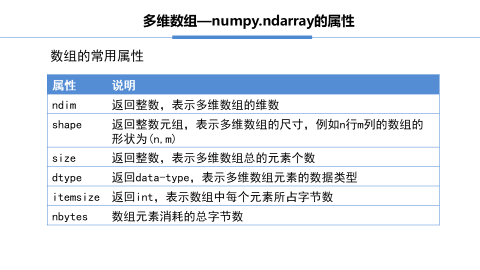
dtype：表示数组所需的数据类型。缺省情况下根据object自动判断。

ndmin：指定生成数组的最小维度。缺省情况下根据object自动判断。

### 2.数组转换为列表有专门的方法:

**ndarry.tolist()**

### 3.参考属性



### 4.arange函数:

**numpy.arange( [start], end, step, dtype=None )**

返回：一维数组。

[start]代表前闭

参数：用法类似 **range**，区别在于各个参数可以是浮点型的。可指定**dtype**，也可由各参数自动决定。

### 5.ndarry 的几种变形方法：

ndarry的**reshape()**方法：  
**ndarray.reshape ( new\_shape )**将多维数组重新**变形**后返回，原始数组保存不变。

ndarray的**shape**属性：  
**ndarray.shape = new\_shape**可直接改变数组的shape属性，数组本身被直接**变形**。

ndarry的**resize()** 方法：  
**ndarray.resize ( new\_shape )**  
直接修改数组的形状，数组本身被直接**变形**。

### 6.linspace 的函数语法：

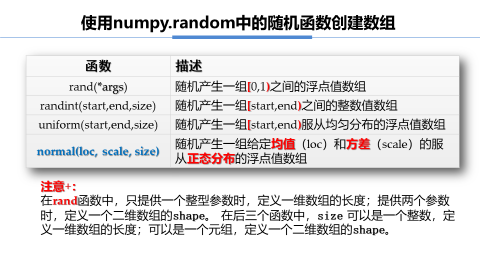
**numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)**

常用参数：

* + **start**：起始的值（整数或者浮点数），闭区间。
  + **end**：结束的值（整数或者浮点数），缺省**闭**区间。
  + **num**：表示在这个区间里生成数字的个数。生成的数组是等间隔的。
  + **retstep**：返回步长

**endpoint**：为**True**时，**end**为闭区间，否则为开区间。

### 7.numpy.random:



### 8.产生正态分布的浮点值随机数

**normal（loc,scale,size）**

* 正态分布（**Normal Distribution**）又称为高斯分布，是一种以均数为中心，左右对称的图形。
* 正态分布有两个参数，即**均值**和**标准差（反映组内个体间的离散程度）**。
  1. **方差**：各个数据与平均数之差的平方的平均数。
  2. **标准差(均方差 scale)**：方差的算术平方根。

### 9.数据类型的转化：

*#返回整个数组数据类型变换后产生的新数组*

arr2=np.float16(arr1) *#原始数组不变*

arr3=arr1.astype(float) *#等效于单独的类型构造器 np.float64()*

*np.int8*

*np.int16*

*np.int32*

*np.int64*

*np.float64*

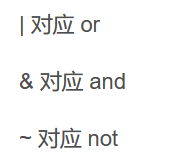
*np.float16*

### 10.where函数：

*#where函数返回数组中值为True元素的各维下标数组组成的元组*

np.where((a>=5) & (a<=9))

逻辑运算符



### 11.多维数组运算常用函数

****

****

### 12.sort函数

**numpy.sort ( a, axis=-1, kind=None, order=None）**

* a：所需排序的数组
* axis：数组排序时的方向轴
  1. 0：垂直方向排序
  2. 1：水平方向排序
  3. -1：按最后的轴排序（缺省值）
  4. None：全部展开后排序成一维数组
* kind：数组排序时使用的方法：
  1. “quicksort”：快排（缺省值）
  2. “mergesort”：混排
  3. “heapsort”：堆排
  4. “stable”：稳定排
* order：提供一个字符串或列表，当a是定义了字段的数组时，此参数列表指定按第一个、第二个……字段进行排序。

**numpy.argsort ( a, axis=-1, kind=None, order=None）**

参数：与sort函数相同

返回：排序后的索引值（下标）

## 第二部分：pandas模块

### 1.创建Series对象语法：

**pandas.Series(data=None, index=None, dtype=None, name=None, ……)**

主要参数：

data：数据源。可以是列表、字典或者numpy的一维ndarray对象

index：设置索引名，可以是以类似列表表示的索引名，缺省值为索引号。

dtype：可以是numpy的dtype，缺省状态为根据data的数据自动决定。

name：给 Series 一个名字。在后续最重要的 DataFrame中可表现为一列的列名(列索引名)或一行的行名(行索引名)。

### 2.日期函数：

生成日期的DatetimeIndex对象

**pandas.date\_range(start=None, end=None, periods=None, freq=None, ……)**

主要参数：

* start：起始日期。以字符串或时间日期数据提供。
* end：结束日期。以字符串或时间日期数据提供。
* periods：生成数据的个数，
* freq：生成频率。以字符串形式提供的时间单位。比如“H”、“D”、“W”、“M“

### 3.Series增减元素：

**Series.append(to\_append, ignore\_index=False,……)**

该方法对象不支持直接添加对象元素（与list的append不同），可以通过该方法将 to\_append对象添加到 Series 对象尾部，此方法返回一个新的、被添加过元素的 Series 对象，原Series对象内容不变。

常用参数：

* to\_append：单个的Series或者是Series组成的列表或元组。
* ignore\_index：忽略原有索引名

-False：添加后，保持原有索引名，如果没有索引名的都会将索引号转成索引名后再添加(缺省值)。

-True：忽略所有已有的索引名，将结果中新的索引号转成索引名。

**Series.drop(labels=None, inplace=False……)**

可以删除一个或多个对象元素的方法。

常用参数：

* labels：需删除元素的索引。单一的索引，或离散的放入列表的索引。
* inplace：

-True：则在Series内直接删除，原Series被改变。

-False：不改变原对象的内容，返回一个新的被删除过元素的Series对象。

### 4.Dataframe创建

**pandas.DataFrame(data=None, index=None, columns=None, ……)**

返回：DataFrame对象。

常用参数：

* data：二维表格中的具体数据，通常由ndarray的二维数组、字典、列表等可迭代的数据、或者另外的DataFrame对象构成的数值。
* index：指定行索引名，可以由ndarray的一维数组对象、列表等构成，其中每个元素可以是指定的字符串、整数等。缺省值为行索引号。通常不用指定。
* columns：指定列索引名（列名），其余和index参数相似。通常需要指定，二维表格表头由此决定。

### 5. 使用字典修改列索引名

df1=df.rename(columns={"A": "a", "B": "c"})

### 6.读取CSV文件函数的常见格式

**pandas.read\_csv(file, sep=',', header='infer', names=None,**

**encoding=None, index\_col=None, usecols=None, nrows=None……)**

**返回：DataFrame。**

**常用参数：**

* file：文件名或文件句柄，甚至可以是URL。
* sep：指定分隔符，默认为“,”
* header：指定表头列名所在的行。默认值为 'infer'（推断）列名：如果 names 参数未设定列名，则从文件的首行推断出列名；如果 names 参数设定了列名，则使用之，代表文件中无表头，只有纯数据。如果 header 和 names 都有设定，header 优先。
* names：表头的列名，可以用类似列表来表示多个列名。
* encoding：文件编码格式，默认为"utf-8”。中文系统中ANSI类型的编码应设置为"gbk"。
* index\_col：数据读入后，指定将作为行索引名的列号或列名。用文件中某些列整列的内容当作 DataFrame 中的行索引名。缺省值为None。
* usecols：读入指定的列。节约内存。缺省值 None，表示全部列。
* nrows：需读入的行数。节约内存。缺省值 None，表示全部行。

### 7. 读取 Excel 文件函数的常见格式：

**pandas.read\_excel(io, sheet\_name=0, header=0, names=None,**

**index\_col=None, usecols=None, nrows=None, engine=None,……)**

返回：如果读取单张工作表，返回 DataFrame，如果是多张，则返回一字典，键为工作表索引名，值为 DataFrame。

比 read\_csv 函数功能更为强大，参数更多。

常用参数(其它常用参数功能与 read\_csv 函数类似)：

* io：文件名、或文件句柄，甚至可以是URL。
* sheet\_name ： 缺省值：为 0，获取首个工作表。单个或多个工作表：可以指定单个或以列表方式指定多个工作表的表索引号或表索引名。
* None： 代表所有工作表。
* engine：选择读取 excel 文件的专用模块。“xlrd”, “openpyxl” 或 “odf”

### 8. DataFrame输出至CSV(excel)文件+

**DataFrame.to\_csv(path\_or\_buf = None, sep = ',', columns = None, header = True, index = True, index\_label = None, encoding = None,……)**

返回：如果 path\_or\_buf 为 None，则将返回的 csv 格式作为字符串返回。否则返回 None。

常用参数：

path\_or\_buf ：文件名或文件句柄。

### 9. iloc 属性索引号（位置）访问DataFrame对象+loc 属性索引名访问DataFrame对象

**(1)** **代表位置的索引号。 格式：**

**DataFrame.iloc [ 行索引号 , 列索引号 ]**

**行与列的索引号可以为下列几种表现形式：**

**单一索引号**

**离散索引号列表（或类似列表）**

**切片索引号（头闭尾开）**

**(2) DataFrame.loc [ 行索引名 , 列索引名 ]**

**行与列的索引名可以为下列几种表现形式：**

**单一索引名**

**离散索引名列表（或类似列表）**

**切片索引名（头尾全闭）**

### 10. 条件筛选访问DataFrame对象

上述三种访问 DataFrame 的手法中，行和列的定义还可以分别接受逻辑值列表，达到筛选的效果，类似数组中的筛选方式。但与数组不同的是：行、列都可以有各自的逻辑值列表。

DataFrame [ 行逻辑值列表 ]

DataFrame.loc [ 行逻辑值列表 , 列逻辑值列表 ]

DataFrame.iloc [ 行逻辑值列表 , 列逻辑值列表 ]

行逻辑值列表：元素个数等于行数的一维逻辑值列表或数组或 Series。

列逻辑值列表：元素个数等于列数的一维逻辑值列表或数组或 Series

### 11. 字符串转整型, 字符串转np.datetime64类型

*对所有元素进行字符串的处理，必须使用str.方法/str[…]*

w.最高气温=w.最高气温.str[:-1]

*#Series有astype方法，转换成 np的数据类型*

w.最高气温=w.最高气温.astype(np.int8)

w.日期=**pd**.**to\_datetime**(w.日期) *#转成 np.datetime64[ns]类型*

### 12. 查找重复的数据行 , 删除重复的数据行

**DataFrame.duplicated(subset = None, keep = 'first')**

返回：逻辑值的 Series，在重复的行对应位置设置重复标记为 True。

参数：

* subset：列名或列名序列，指定需要判定是否有重复值的列，默认None，判定所有的列。
* keep：'first'，'last'，False。决定在返回的Series中打重复标记的方式，有重复时，哪一个不打重复标记。

-"first"：将重复的行标记置为 True，第一个重复的行不标记，保存原样（缺省）。

-"last"：将重复标的行记置为 True，最后一个重复的行不标记，保存原样。

-False：所有的重复行都标记为 True。

**DataFrame.drop\_duplicates(subset=None, keep='first', inplace=False, ignore\_index=False)**

返回：删除重复行后的DataFrame 或 None。

参数：

* subset：列名或列名序列，指定需要判定是否有重复值的列，默认None，判定所有的列。
* keep：'first'，'last'，False。决定重复后保留的行。

-"first"：删除重复的行，第一个重复的行保留（缺省）。

-"last"：删除重复的行，最后一个重复的保留。

-False：删除所有的重复行。

* inplace: 略（同前）。
* ignore\_index：为 True 时，重设结果 DataFrame 的行索引名。

### 13. 删除缺失数据

**DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)**

该方法可以删除缺失数据所在的行或列。

返回：视 inplace 参数而定。

参数：

* axis：删除行或列。 0 删除行，1 删除列。
* how:“any”：表示删除只要存在NaN的行或或列。此为默认值。

“all”：表示删除全部值都为NaN的行或列。

* thresh：至少留下有效数据大于或等于 thresh 值的行或列。
* subset：index 或 column 列表，按行列设置子集，在子集中查找缺失数据。
* inplace：略（同前）。

### 14. 填充缺失数据

**DataFrame.fillna(value=None, method=None, axis=None, inplace=False, ……)**

该函数可以用某些特定的值填充缺失数据所在的元素。

返回：视 inplace 参数为而定。

常用参数：

* value：填充值，可以是标量(简单数据类型的数据)、dict、Series 或 DataFrame。
* method：填充的方法。

-"pad" 或 "ffill"：使用同列(行)前一行(列)的值填充。

-"backfill" 或 "bfill"：使用同列(行)后一行(列)的值填充。

-None：使用 value 参数的值。(缺省)

* axis：0 沿着列填充，1 沿着行填充。
* inplace：同前。（略）

### 15. 使用concat函数合并数据

**pandas.concat(objs, axis=0, ignore\_index = False,……)**

可以合并 Series 和 DataFrame 类型。

返回：合并后的新对象。不影响参加合并的各个原始对象。

常用参数：

* objs：各个参与合并的对象所组成的列表。
* axis：合并方向，0为合并行（上下合并），1为合并列（左右合并）。
* ignore\_index：合并时忽略各数据对象的原始行索引名，重新设置新的行索引名为新的行索引号

### 16.Drop函数

**DataFrame.drop(labels=None, axis=0, index=None, columns=None, inplace=False,……)**

该函数可以根据索引名，删除 DataFrame 的行或列。

返回：视 inplace 参数而定。

常用参数：

* labels：索引名或多个索引名组成的列表。由 axis 决定是行索引名还是列索引名
* axis：删除方向，0 为删除行，1 为删除列。行索引名默认为 0 。
* index、columns：直接指定需删除的行索引名或列索引名。
* inplace：略（同前）**。**

### 17. DataFrame 数据的排序

**DataFrame.sort\_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na\_position='last', ignore\_index=False)**

功能：根据关键字，按指定轴向排序行或列。

DataFrame 的排序与数组排序效果大相径庭。

当数组按 0 方向排序时，会破坏行的完整性，按 1 方向排序时，会破坏列的完整性。

DataFrame 按 0 方向排序，即行与行之间的先后顺序发生了变化，但每一行的数据排序前和排序后是相同的。也可以按 1 方向排序，效果相似，但排的是列。

最常用的排序方案：按 0 方向排，即表格数据有多个列，可以设置按某个关键字（列名）或多个关键字升序或降序排序**。**

参数：

* by：将列名或列名组成的列表设为排序关键字，按指定的关键字排序。
* axis：排序的轴方向。0 决定行与行的上下顺序；1 决定列与列的左右顺序。
* ascending：True 升序、False 降序。
* inplace：略（同前）。
* kind：排序算法“quicksort”、“mergesort”, “heapsort”。
* na\_position：缺失值参加排序时的固定位置"first"(置**顶)**、"last"(沉底)。
* ignore\_index：结果中去除原先行或列的索引名

### 18.groupby and agg

**DataFrame.groupby(by, axis=0, as\_index= True, sort= True, ……)**

将 DataFrame 对象分为若干组，并返回一个 DataFrameGroupBy 对象。

常用参数：

* by：分组依据（key），通常为列索引列表，称为关键列。
* axis：分组的轴方向。0 一个小组 n 行，1一个小组 n 列。默认 0。
* as\_index：表示聚合标签是否以索引形式输出，默认True。
* sort：表示是否对分组后各小组间的顺序按 key 排序。默认True。

返回一个DataFrameGroupBy对象，将该对象想象成整个 DataFrame (缩写 DF )被分组后的多个小的 DF，它是可迭代的，可用迭代方式获取分组的“键”以及该键所属的小组 DF。

聚合方法

**GroupBy.agg(func, \*args,\*\*kwargs)**

返回：根据提供的统计函数统计，并聚合，以 DataFrame 返回结果。

常用参数：

* func：提供统计函数。既可以是 python 中的统计函数，也可以是 numpy 模块中的统计函数，甚至可以是自定义函数。各种提供形式如下：

统计函数名：相当于回调函数形式。比如 sum、mean

统计函数名的字符串表示：比如："sum"、"mean"

上述两种类型的多个函数组成的列表：比如：[np.cumsum,'mean']

字典：以需统计列索引名为键，统计函数为值（可以是上述类型）的字典

### 19. 求最值对应的索引名

pandas.Series.idxmax()返回的是Series中最大值的索引名

pandas. DataFrame.idxmax()返回的是DataFrame中每一列最大值的索引名

pandas.Series.idxmin()返回的是Series中最小值的索引名返回

pandas. DataFrame. .idxmin()返回的是DataFrame中每一列最小值的索引名

## 第三部分：matplotlib模块

**import matplotlib.pyplot as plt**

### 1.绘制折线图

**matplotlib.pyplot.plot(x, y, format\_string, ……)**

在当前画布 figure 的当前绘图区 Axes 中根据提供的 x，y 中的数据，在坐标系中一一对应绘制各点，以及上一点与下一点之间的连续线条。（折线图）

返回：由 matplotlib.lines.Line2D 线条组成的列表。

常用参数：

x：x 轴数据，以类似数组的数据提供。

y：y 轴数据，以类似数组的数据提供。

format\_string：控制点与线的格式字串 ，由表示色彩、点的形状以及线条线型的字符组合而成。



### 2.保存图形

**matplotlib.pyplot.savefig(fname, dpi=None, quality=None，……)**

保存当前画布内容至图像文件，或者至后台相关的对象中。

常用参数：

fname：字符串文件名或者已打开的文件对象。如果 format 参数没有被设置，则输出格式从扩展名中获得（缺省：“PNG”）。  
支持的图像文件格式：eps, jpeg, jpg, pdf, pgf, png, ps, raw, rgba, svg, svgz, tif, tiff

dpi：分辨率。以每英寸点数为单位。如果为 None，则默认为使用当前画布图像的 dpi 值。

quality：图像质量。范围从1（最差）到95（最佳）。仅当格式为 jpg或 jpeg 时适用，否则忽略。如果为 None，则默认为（默认值：95）。应避免超过 95 的值；100 则完全禁用 JPEG 优化。

### 3. 创建画布

**matplotlib.pyplot.figure(num=None, figsize=None, dpi=None, facecolor=None, ……)**

返回：Figure 画布对象。

常用参数：

num：画布的编号。如为 None，则系统自动提供（从 1 开始）。如为字符串，则作为画布窗口中的标题（朱庇特中目前无此效果）。

figsize：以英寸为单位的画布的尺寸，以 (宽, 高 ) 方式提供。缺省值为 (6.4, 4.8)。

dpi：图像分辨率，设置每英寸的点数。缺省值为 100dpi。

facecolor：画布的颜色。

### 4. 创建子图

**matplotlib.figure.Figure.add\_subplot(nrows, ncols, index, \*\*kwargs)**

创建的子图将在具有 nrows 行和 ncols 列的虚拟布局网格上占据索引位置。索引从左上角的 1 位置开始并向右向下逐渐增加。

返回：创建的子图，类型为 axes.SubplotBase 或者其它Axes的子类。

一次只能创建一个子图。

常用参数：

nrows：在画布中的虚拟布局子图行数。

ncols：在画布中的虚拟布局子图列数。

index：在虚拟布局中创建的子图的索引号。决定子图的具体位置。

可以提供一个三位的数字，以合并的方式决定 nrows、ncols 和 index，因为通常行数和列数不会超过 9。

也可以三个参数分开提供。

### 5. 创建画布和子图。

**matplotlib.pyplot.subplots(nrows=1, ncols=1, sharex=False, sharey=False, num=None ……)**

创建一张画布和一组子图。

返回：一个两个元素的元组。

* 1. fig：Figure （画布）
  2. ax：Axes 对象（子图）或 Axes 对象数组（多个子图）

常用参数：

nrows，ncols：可选，默认值：1。子图网格的行数/列数。

sharex，sharey：逻辑值或 'none'，'all'，'row'，'col'。控制 x(sharex)或 y(sharey)轴之间的属性共享：

True 或 "all"：x 轴或 y 轴将在所有子图中共享。

False 或 "none"：每个子图的 x 轴或 y 轴将是独立的。

"row"：每个子图行将共享一个 x 轴或 y 轴。

"col"：每个子图列将共享一个 x 轴或 y 轴。

当子图沿列具有共享的 x 轴时，仅创建底部子图的 x 刻度标签。同样，当子图沿行具有共享的 y 轴时，仅创建第一列子图的 y 刻度标签。若要稍后打开其他子图的刻度标签，可使用 tick\_params。

num：指定画布编号。在指定的画布上创建子图**。**

### 6.绘制散点图。

**Axes.scatter (x, y, s=None, c=None, marker=None, alpha=None, linewidths=None, edgecolors=None,…… \*\*kwargs)**

功能：绘制散点图。

常用参数：

x, y：散点图的数据源。类数组数据。

s 、c：点标记的尺寸（正常标记尺寸的平方）、标记的颜色。

marker：标记的风格，

alpha：0 透明，1 不透明。

linewidths：标记边缘线的宽度。

edgecolors：标记边缘线的颜色。

### 7.绘制直方图函数

**Axes.hist(x, bins=None, range=None, histtype='bar', label=None, ……\*\*kwargs)**

根据数据绘制直方图。

常用参数：

x：指定要绘制直方图的数据。

bins：指定直方图条形的个数，默认 10。或序列表示的个区域边界。

range：指定直方图数据的上下界，超出范围的将被忽略。默认包含绘图数据的最大值和最小值。

histtype：画图的形状。

'bar', 'barstacked', 'step', 'stepfilled'。

返回值：一个三个元素的元组，三个元素分别为：

n：表示每一区间的数据个数，并组成序列。

bins：区间分界点组成的序列。个数为 n 中的个数+1。

patches：各个分区对象组成的序列。

### 8.绘制饼图

**matplotlib.pyplot.pie(x, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=None, radius=None, counterclock=True,……)**

常用参数：

x：每块扇形占总和的百分比数值，如果sum(x) > 1会使用sum(x)自动计算每块占总的百分比。

explode：每块扇形被炸离中心的距离，以相对于半径的比例来指定。

labels：每块扇形外侧显示的说明文字（数据标签）

colors：每块扇形的默认颜色=(‘b’, ‘g’, ‘r’, ‘c’, ‘m’, ‘y’, ‘k’, ‘w’)

autopct：控制饼图内百分比文字格式设置,可以使用格式化字符串。

pctdistance：饼内文字离开中心的距离，以相对于半径的比例来指定autopct的位置刻度,默认值为0.6

shadow：在饼图下面画一个阴影。默认值：False，即不画阴影；

labeldistance：label 标签的绘制位置，类似于 pctdistance，相对于半径的比例，默认值为1.1, 如<1则绘制在饼图内侧

startangle：起始绘制角度,默认图是从x轴正方向逆时针画起。

### 9.柱状图函数

**9.1 matplotlib.pyplot.bar(x, height, width=0.8, bottom=None, yerr……\*\*kwargs)**

**常用参数：**

x：x 轴数据。通常，每个分类与连续的各个整数对应，给 height在 x 轴上定位。

height：y 轴对应柱的高度。数据个数与 x 的数据个数相同。

width：x 轴对应柱的相对宽度（等分宽度中的占比 ），默认 0.8（保证柱与柱之间留有0.2的空隙）。

bottom：y 轴基准值，默认为 0。

yerr：y轴误差线数据。

**9.2 用 pandas 的绘图功能绘制柱状图**

**DataFrame.plot.bar (x=None, y=None, \*\*kwargs)**

**常用参数：**

x：各个分类的名字，如果不指定，则使用 DataFrame的行索引名作为各个分类名。

y：各个分类的数值数据，如果不指定，则使用 DataFrame 的所有数值列数据，列名为图例中的数据名。

rot：轴刻度文字的角度。

xlabel、ylabel：x 和 y 轴的轴标签。

xlim、ylim：以2个元素的元组或列表决定 x 或 y 轴的上下限。

title：子图标签。

grid：逻辑值，表示是否显示网格线（x、y 轴同时控制）

figsize：以英寸为单位的画布的尺寸，以 (宽, 高 ) 方式提供。缺省值为 (6.4, 4.8)。

### 10.多项式拟合

**numpy.polyfit(x, y, deg, rcond=None, full=False, w=None, cov=False)**

返回：通过 *y=f(x)* 关系中大量 x 和 y 的对应数据，以数组类型返回拟合后的多项式系数。

常用参数：

x, y：多项式 *y=f(x)* 中大量实际 x 值和 y 值，以类似数组方式提供。

deg：设定多项式中所希望的最高阶数。

**numpy.poly1d(c\_or\_r, r = False, variable=None)**

根据多项式系数，封装并返回多项式对象，以便进一步根据多项式计算更多的 x 值所对应的 y 值。

返回：多项式对象。如要算出真正的多项式的值，需要将多项式对象当作函数使用，参数可以是多个 x 值的序列，可得到多个多项式的结果值。

常用参数：

c\_or\_r ：多项式的系数，类数组型。以幂次递减，或者如果 r 参数的值为True，则表示为多项式的根（多项式的值为 0 时 x 的值）。例如:

poly1d([1, 2, 3]) 代表 *x2 + 2x + 3*

poly1d([1, 2, 3], True) 代表 *(x-1)(x-2)(x-3)= x3 - 6x2 + 11x -6*

r ： 布尔型，可选 如果为 True，则 c\_or\_r 指定多项式的根。默认值为 False。

多项式对象属性：

c：多项式的各个系数

r：多项式的根（多项式的值为 0 时 x 的值）

o：多项式的阶数（最高幂次数）

### 11.legend

**matplotlib.pyplot.legend(labels, loc, bbox\_to\_anchor, title, fontsize，…… )**

实用格式：

legend()

legend(labels)

legend(handles, labels)

常用参数：

handles：已绘制的各种图形实例。可以用类似列表的方式提供。缺省值为所有已绘制的图形。

labels：图例文字。可用类似列表的方式提供。缺省值为绘制图形时在绘制的方法或函数中所提供的 label 参数。

loc：相对位置。它和 bbox\_to\_anchor 参数协同配合决定图例的最终位置。

当不提供 bbox\_to\_anchor 参数时，loc 决定图例在图形边界中出现的相对位置。

当 loc 为两个元素的元组时，表示图例的左下角的 x,y 坐标(在整个坐标长度中的占比)，此时 bbox\_to\_anchor 参数将失效。

通常可以用字符串或数字来表示在区域中的相对位置。

## 第四部分:机器学习

#### 涉及到的模块

**sklearn.model\_selection**

**sklearn.metrics**

**sklearn.neighbors**

**sklearn.linear\_model**

### 数据集的拆分

**sklearn.model\_selection.train\_test\_split(\*array, test\_size=0.25, train\_size, random\_state=None, shuffle=True, ...)**

功能：按照比例将多个数据集每个拆分为两个数据集（训练集和测试集）。

常用参数：

\*array：多个相同长度的批量数据集，可以是 list、ndarray、DataFrame 等。通常给出特征数据集和标签数据集两个批量数据。

test\_size：测试样本占比，如果是整数的话就是样本的数量。

train\_size：训练样本占比，如果是整数的话就是样本的数量。通常该数值不指定，自动补齐 test\_size 所剩下的数据。

random\_state：是 shuffle 执行时随机数的种子。如果每次需要不一样的数据， 可以将 random\_state 设置为缺省。

shuffle：逻辑值，是否在划分前要进行随机洗牌操作。缺省为 True。

返回：每个数据集被分成两部分形成的一个列表，通常为

[ X\_train, X\_test, y\_train, y\_test ]

### 2.分类评价指标

****

### 3.回归评价指标

****

### 4.KNN算法

**KNeighborsClassifier对象及常用方法**

#### （1）创建模型对象

**sklearn.neighbors.KNeighborsClassifier(n\_neighbors=5, weights=’uniform’, algorithm=’auto’, leaf\_size=30,p=2, metric=’minkowski’, metric\_params=None, n\_jobs=None, \*\*kwargs)**

返回： KNeighborsClassifier 对象实例。

常用参数：

* n\_neighbors：int 型,可选，缺省值为 5，就是 KNN 中的近邻数量 k 值。
* weights：计算距离时使用的权重。
  + 缺省值是“uniform”，均匀的权重。每个邻域中的所有点的权重均相等。
  + 也可以取值“distance”，权重点按其距离的倒数表示，查询点的近邻比远邻具有更大的影响力。
  + 用户定义的函数，它接受距离数组，并返回包含权重的相同形状的数组。
* metric：距离的计算。
  + 缺省值是“minkowski”。当 p=2, metric=’minkowski’时，使用的是欧式距离。
  + **p=1,metric=’minkowski’时为曼哈顿距离。**

#### （2）训练

* **fit(X,y)：训练/拟合（自动调整模型参数得到最终模型对象）。**
  + **X：训练集特征数据样本。（二维数据）**
  + **y：训练集的目标值（标签）数据样本。（一维或二维数据）**
  + **返回：KNeighborsClassifier 对象实例。**

#### （3）预测

* **predict(X)：对测试集进行预测。**
  + **X：测试集特征数据样本。**
  + **返回：根据给定的特征数据预测得出其所属的类别标签（测试集目标数据）。**

#### （4）评估

* **score(X, y, ……)：得分指标，全部类别平均正确率。**
  + **X：测试集特征数据样本。**
  + **y：测试集中的目标真实值。**
  + **返回：各类别平均正确率(mean accuracy)。**

#### （5）邻居信息

* **kneighbors(X=None, ……)：返回指定的邻居信息。**
  + **X：测试集特征数据样本。**
  + **n\_neighbors：最近邻居数。**
  + **return\_distance：逻辑值。返回邻居的距离和邻居的索引号，或只返回邻居的索引号。**
  + **返回：由 return\_distance 决定。**

### 5.KNN交叉验证得分

**sklearn.model\_selection.cross\_val\_score(estimator, X, y=None, \*, groups=None, scoring=None, cv=None, n\_jobs=None, verbose=0, fit\_params=None, pre\_dispatch='2\*n\_jobs', error\_score=nan)**

使用 K 折交叉验证测试模型的得分，通常用来观察并选择模型的“超参数”。

常用参数：

estimator：估算器，即各种有标准 fit 、predict 方法的模型对象。

X：整个特征数据集（将会被自动交叉划分）。

y：整个目标数据集（将会被自动交叉划分）。

cv：K 折交叉验证所用的 K。缺省值为 5 。

返回：每折训练的得分（数组），默认使用 estimator 模型对象本身的 score 方法。不返回训练后的模型或影响已有的模型。

### 6.归一化和标准化

#### （1）归一化

**创建 MinMaxScaler 对象:**

sklearn.preprocessing.MinMaxScaler(feature\_range=(0, 1), \*, copy=True……)

通过将每个要素缩放到给定范围来变换元素。

返回：MinMaxScaler 对象实例。

常用参数：

feature\_range：表示所需的转换数据范围。元组，默认为(0,1)。

**MinMaxScaler 对象的方法:**

fit(x)

x：根据实际数据 x，计算用于以后缩放的用的最小值和最大值，并将其存放于 MinMaxScaler 实例的属性中。

返回：被 fit 过的实例。通常无需返回值，因为实例本身已经发生了变化(属性被赋值)。

transform(x)

x：根据实例内的已经 fit 的最大最小值属性，转换具体的数据 x。

返回：x 转换后的结果。

fit\_transform(x)

x：直接根据实际数据 x，先 fit 确定最大最小值，然后对实际数据再进行转换。

返回：转换后的数据（类似数组）。

#### （2）标准化

**创建StandardScaler 对象**

sklearn.preprocessing.StandardScaler(\*, copy=True, with\_mean=True, with\_std=True…)

缩放到标准差来标准化特征。

返回：StandardScaler 对象实例。

常用方法：

fit、 transform、fit\_ transform， 类似 MinMaxScaler 对象的方法。

### 7.线性回归

LinearRegression 对象及常用方法

#### （1）创建模型对象

**sklearn.linear\_model.LinearRegression(……)**

创建最小二乘法回归分析模型对象。

**常用参数**：通常可以不指定**超参数**。

**返回**： **LinearRegression** 对象实例。

**常用属性**：

**coef\_**：除截距 **𝜃0**外的所有系数（数组）。**𝜃1 ~ 𝜃n**

**intercept\_**：截距 **𝜃0** 。

#### （2）训练

**fit(X, y, sample\_weight=None) ：拟合线性模型，模型实例本身已被改变。**

X：训练集特征值样本(类数组)，二维数据。

y：训练集目标值(类数组)。可以一维，也可以是二维数据。

返回：练成后的对象实例，通常无需保存返回值至变量，因为模型实例已经练成。

#### （3）预测

**predict**(**X**)：使用线性模型进行预测。

X：测试集特征值样本(类数组)，二维数据。

返回：预测值(数值) 。

#### （4）评估

* **score**(**X**, **y**, **sample\_weight=None**)：得分
  + **X**：测试集特征值样本(类数组)，二维数据。
  + **y**：测试集中真实目标值(类数组)。可以一维，也可以是二维数据。
  + **返回**：***R2***指标，最佳为 **1**，差的可能为**负**数。