《Python数据分析案例教程》

第十一章 综合案例

综合案例

Comprehensive case

本章综合运用前面章节数据分析相关知识，按照数据分析的基本步骤，由浅入深设计实现两个相对完整的数据分析综合案例。第一个案例侧重数据准备和探索性数据分析，第二个案例在探索性数据分析的基础上，更注重运用数据挖掘和机器学习算法完成部分验证性数据分析任务。

目录 content

PART ONE

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

任务描述

本案例使用**Python**爬取百度百科网站关于《乘风破浪的姐姐》综艺节目嘉宾的网络数据，并进行可视化数据分析。

百度百科网站（网址https://baike.baidu.com/item/乘风破浪的姐姐）保存了节目中30位选手的详细资料及相关数据。

本案例首先爬取并解析百度百科网站关于该综艺节目嘉宾的背景数据，然后使用**扩展库matplotlib相关函数**进行探索性数据分析。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

任务描述

本案例数据获取的流程：

使用Python编写爬虫程序

模拟用户登录浏览器的行为

向目标站点发送数据请求，并接收来自目标站点的响应数据

最后提取其中有用的数据并保存到本地数据文件中

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

任务描述

数据爬取过程主要包含如下步骤：

01 发起请求

向百度百科网站发起HTTP请求，也就是发送一个Request。该请求可以包含链接、请求头、超时设置等信息，等待服务器响应。

本案例使用Python中简单易用的HTTP库Request，通过requests.get(url)发送一个HTTP GET请求，等待服务器返回响应内容。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

任务描述

如果服务器能正常响应用户请求，返回的网页内容会保存为一个Response对象，可以是HTML文件、json字符串、二进制数据（图片或者视频）等，其内容包含需要获取的页面数据。

02 获取响应内容

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

任务描述

03 解析内容

响应内容如果是HTML格式的数据，可以用正则表达式，或者页面解析库进行解析；如果是json格式的数据，可以直接转换为json对象解析；如果是二进制数据，可以直接保存或者进一步处理。

本案例使用Python扩展库BeautifulSoup从HTML或XML文件中提取数据。BeautifulSoup支持Python标准库的HTML解析器，还支持一些第三方解析器，如lxml。Python标准库的HTML解析器执行速度适中、文档容错能力比较强，使用方法为：BeautifulSoup(markup, "html.parser")；第三方库lxml的HTML解析器执行速度快、文档容错能力强，需要安装C语言库，使用方法为：BeautifulSoup(markup, "lxml")。一般推荐使用lxml作为解析器。本案例为了获得效率更高的数据解析效果，采用lxml作为解析器。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

任务描述

数据的保存形式多样，可以保存为文本，也可以保存到数据库，或者保存为特定格式的文件，本案例将获取的有效数据保存为json文件。

04 保存数据

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

本案例需要收集的数据包括参赛选手信息和每个选手的百度百科页面信息等。

按照数据爬取的主要步骤，首先导入可能用到的标准库和扩展库。

```
In[ ]:import json
      import re
      import requests
      import datetime
      from bs4 import BeautifulSoup
      import os
```

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

首先登陆百度百科网站，爬取综艺节目《乘风破浪的姐姐》所有参赛选手的信息，返回页面数据。自定义爬取函数crawl_wiki_data()的主要代码如下：

01
爬取参赛选手页面数据

```
def crawl_wiki_data():
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36'}
    url='https://baike.baidu.com/item/乘风破浪的姐姐'
    try:
        response = requests.get(url,headers=headers)
        soup = BeautifulSoup(response.text,'lxml')
        tables = soup.find_all('table')
        crawl_table_title = "按姓氏首字母排序"
        for table in  tables:
            #对当前节点前面的标签和字符串进行查找
            table_titles = table.find_previous('div')
            for title in table_titles:
                if(crawl_table_title in title):
                    return table
    except Exception as e:
        print(e)
```

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

```
def crawl_wiki_data():
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36'}
    url='https://baike.baidu.com/item/乘风破浪的姐姐'
```

01

爬取参赛选手页面数据

代码第2行用于定义请求头部：服务器通过请求头部的用户代理信息（User Agent）来判断一个请求的发送者是正常的浏览器还是爬虫。为了防止被服务器的反爬虫策略禁止，这里为请求添加一个HTTP头部来伪装成正常的浏览器。因此将一个用户代理信息传递给headers字典变量。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

01
爬取参赛选手页面数据

代码第6行用于发送网络请求：这里使用requests的get()函数获取目标网站百度百科中《乘风破浪的姐姐》综艺节目首页。获取网页最常用的get()函数将请求的数据，包括请求的头部信息都放在网址url中，通过浏览器发送消息给网址所在的服务器。

```
try:
    response = requests.get(url,headers=headers)
    soup = BeautifulSoup(response.text,'lxml')
```

代码第7行将请求返回的响应传入解析器：创建BeautifulSoup对象，将需要解析的内容即response.text传入lxml解析器，这里response.text字符串存放了获取的目标网页源代码。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

01

爬取参赛选手页面数据

```
    tables = soup.find_all('table')
    crawl_table_title = "按姓氏首字母排序"
    for table in  tables:
        #对当前节点前面的标签和字符串进行查找
        table_titles = table.find_previous('div')
        for title in table_titles:
            if(crawl_table_title in title):
                return table
except Exception as e:
    print(e)
```

代码第8~15行爬取节目嘉宾数据：首先使用find_all()方法找到所有的“table”标签，再通过find_previous()方法找到当前节点前面的“div”标签，接着找到“按姓氏首字母排序”字符串所在的表格（“table”标签），从而找到两季节目参赛选手的页面数据所在位置。使用print函数输出变量table_titles和table的页面数据，部分内容分别如下页图所示。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

01

爬取参赛选手页面数据

变量table_titles和table对应的部分页面内容

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

```
<div class="para-title level-3" label-module="para-title">
<h3 class="title-text"><span class="title-prefix">乘风破浪的姐姐</span>第一季</h3>
</div>
<div class="para-title level-3" label-module="para-title">
<h3 class="title-text"><span class="title-prefix">乘风破浪的姐姐</span>第二季</h3>
</div>
<div class="para" label-module="para">*（按姓氏首字母排序）</div>
ok
<div class="para" label-module="para">*（按姓氏首字母排序）</div>
ok
<div class="para" label-module="para"><b>第一季</b></div>
<div class="para" label-module="para"><b>第二季</b></div>
<div class="para" label-module="para"><b>第一季</b></div>
<div class="para-title level-3" label-module="para-title">
<h3 class="title-text"><span class="title-prefix">乘风破浪的姐姐</span>第一季</h3>
</div>
<div class="para-title level-3" label-module="para-title">
<h3 class="title-text"><span class="title-prefix">乘风破浪的姐姐</span>第二季</h3>
</div>
<div class="para-title level-2" label-module="para-title">
<h2 class="title-text"><span class="title-prefix">乘风破浪的姐姐</span>获奖记录</h2>
<a class="edit-icon j-edit-link" data-edit-dl="6" href="javascript:;"><em class="cmn-icon wiki-lemma-icons wiki-lemma-icons_edit-lemma"></em>编辑</a>
</div>
<div class="para-title level-3" label-module="para-title">
<h3 class="title-text"><span class="title-prefix">乘风破浪的姐姐</span>播出平台</h3>
</div>
<div class="para-title level-3" label-module="para-title">
<h3 class="title-text"><span class="title-prefix">乘风破浪的姐姐</span>收视率</h3>
</div>
```

01

爬取参赛选手页面数据

print函数输出变量table_titles内容

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

01
爬取参赛选手页面数据

print函数输出变量table部分内容

```
<table data-sort="sortDisabled" log-set-param="table_view"><tr><td align="left" valign="top" width="195"><div class="para" label-module="para"><b><a data-l
emmaid="435383" href="/item/%E9%98%BF%E6%9C%B5/435383" target="_blank">阿朵</a></b></div><div class="para" label-module="para">中国内地女歌手、演员</div><div
class="para" label-module="para">代表作：<a data-lemmaid="67421" href="/item/%E6%B0%B4%E6%9E%9C%E5%A7%91%E5%A8%98/67421" target="_blank">水果姑娘</a>、<a dat
a-lemmaid="83244" href="/item/%E7%86%8A%E7%8C%AB%E5%A4%A7%E4%BE%A0/83244" target="_blank">熊猫大侠</a>、<a data-lemmaid="3900785" href="/item/%E4%B8%93%E5%8
8%97%E4%B8%80%E5%8F%B7/3900785" target="_blank">专列一号</a>等</div><div class="para" label-module="para"><div class="lemma-picture text-pic neweditortablei
mg" style="width:100%;height:auto;max-width:129.96575342466px;">
<a class="image-link" href="/pic/%E4%B9%98%E9%A3%8E%E7%A0%B4%E6%B5%AA%E7%9A%84%E5%A7%90%E5%A7%90/49998987/0/f7246b600c338744ebf8fa670046cef9d72a60590adc?fr
=lemma&amp;ct=single" nslog-type="9317" target="_blank" title="">
<img alt="" class="lazy-img" data-src="https://bkimg.cdn.bcebos.com/pic/f7246b600c338744ebf8fa670046cef9d72a60590adc?x-bce-process=image/resize,m_lfit,w_12
9,limit_1/format,f_auto" src="
F9fX1AAAA0VQI3QAAAAxJREFUeNpiYAAIMAAAAgABT21Z4QAAAABJRU5ErkJggg==" style="width:100%;height:100%;max-width:129.96575342466px;max-height:220px;"/>
</a>
```

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

对爬取获得的参赛选手页面数据进行解析，得到需要的选手信息，包括选手姓名和选手个人百度百科页面链接，保存到work目录下的json文件。为完成此功能，自定义参赛选手页面解析函数parse_wiki_data()，主要代码如下：

```
def parse_wiki_data(table_html):
    bs = BeautifulSoup(str(table_html),'lxml')
    all_trs = bs.find_all('tr')
    stars = []
    for tr in all_trs:
        all_tds = tr.find_all('td')   #tr下面所有的td
        for td in  all_tds:
            #star存储选手信息，包括选手姓名和选手个人百度百科页面链接
            star = {}
            if td.find('a'):
                #找选手名称和选手百度百科链接
                if td.find_next('a'):
                    star["name"]=td.find_next('a').text
                    star['link'] =  'https://baike.baidu.com' + td.find_next('a').get('href')
                elif td.find_next('div'):
                    star["name"]=td.find_next('div').find('a').text
                    star['link'] = 'https://baike.baidu.com' + td.find_next('div').find('a').get('href')
                stars.append(star)
    json_data = json.loads(str(stars).replace("\'","\""))
    with open('work/' + 'stars.json', 'w', encoding='UTF-8') as f:
        json.dump(json_data, f, ensure_ascii=False)
```

02
参赛选手页面数据解析

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

02
参赛选手页面数据解析

代码1-9行

```
def parse_wiki_data(table_html):
    bs = BeautifulSoup(str(table_html),'lxml')
    all_trs = bs.find_all('tr')
    stars = []
    for tr in all_trs:
        all_tds = tr.find_all('td')   #tr下面所有的td
         for td in  all_tds:
            #star存储选手信息，包括选手姓名和选手个人百度百科页面链接
```

自定义函数parse_wiki_data()接收页面爬取函数crawl_wiki_data()的返回值“table”，其中包含着所有选手的页面信息。

将选手页面信息字符串传入lxml解析器，并查找所有“tr”标签及其下面的“td”标签；然后遍历每个“tr”标签下的“td”标签。“td”标签包含的各级“a”标签下对应着节目参赛选手的姓名和代表作等百度百科个人页面链接。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

代码9-23行

```
            #star存储选手信息，包括选手姓名和选手个人百度百科页面链接
            star = {}
            if td.find('a'):
                #找选手名称和选手百度百科链接
                if td.find_next('a'):
                    star["name"]=td.find_next('a').text
                    star['link'] =  'https://baike.baidu.com' + td.find_next('a').get('href'  )
                elif td.find_next('div'):
                    star["name"]=td.find_next('div').find('a').text
                    star['link'] = 'https://baike.baidu.com' + td.find_next('div').find('a').get('href')
                stars.append(star)
```

如果找到了第一个“a”标签，那么将得到的文本内容即选手姓名保存为字典变量的第一个元素，键为“name”，值为该选手的姓名字符串。接下来使用find_next()方法获取与当前元素最接近的“a”标签对应的文本内容，即选手个人页面链接信息，将其作为字典变量的第二个元素，其中键为“link”，值为该选手对应的百度百科个人页面链接网址。反之，如果第一个“a”标签下找到的是“div”标签，继续使用find_next()方法获取与当前元素最接近的“div”标签，找到满足条件的第一个“a”标签节点，获取对应的文本内容，生成star字典变量的前两个元素，即“选手姓名”键值对和“选手个人页面链接网址”键值对。代码第23行将star字典变量中保存的每位选手信息追加至列表变量stars中。至此，变量stars中包含30个字典元素，每个字典元素包含一位参赛选手的信息，即“选手姓名”键值对和“选手个人页面链接网址”键值对。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

02
参赛选手页面数据解析

代码24-26行

```
json_data = json.loads(str(stars).replace("\'","\""))
with open('work/' + 'stars.json', 'w', encoding='UTF-8') as f:
    json.dump(json_data, f, ensure_ascii=False)
```

将转换为字符串的stars变量中可能的“\'”符号统一为“\""符号，然后通过json.loads()函数转换为Python列表数据类型，最后写入work目录下stars.json文件中。至此，函数完成了参赛选手页面数据的解析。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

02
参赛选手页面数据解析

print

out[]: [{'name': '阿朵', 'link': 'https://baike.baidu.com/item/%E9%98%BF%E6%9C%B5/435383'}, {'name': '白冰', 'link': 'https://baike.baidu.com/item/%E7%99%BD%E5%86%B0/10967'}, ...]

使用print语句输出json_data变量中前两个元素的数据

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

根据给定的图片链接列表pic_urls，下载所有选手的图片数据，并保存在以选手姓名命名的文件夹中。自定义图片数据爬取down_save_pic()函数，主要代码如下：

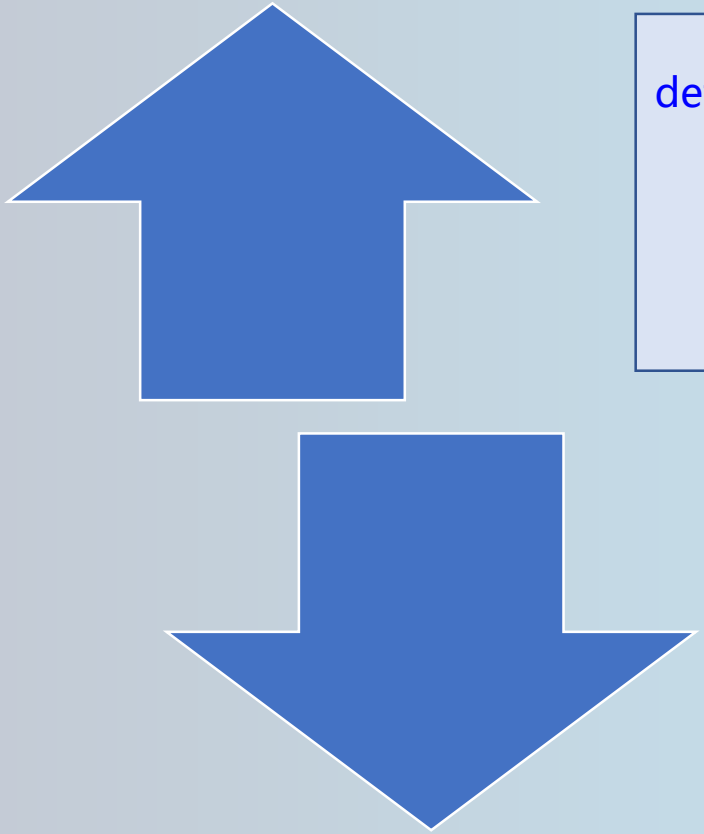
```
def down_save_pic(name,pic_urls):
    path = 'work/'+'pics/'+name+'/'
    if not os.path.exists(path):
        os.makedirs(path)
    for i, pic_url in enumerate(pic_urls):
        try:
            pic = requests.get(pic_url, timeout=15)
            string = str(i + 1) + '.jpg'
            with open(path+string, 'wb') as f:
                f.write(pic.content)
                #print('成功下载第%s张图片: %s'% (str(i+1), str(pic_url)))
        except Exception as e:
            #print('下载第%s张图片时失败: %s'% (str(i + 1), str(pic_url)))
            print(e)
            continue
```

03
爬取选手图片数据

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取



```
def down_save_pic(name,pic_urls):
    path = 'work/'+'pics/'+name+'/'
    if not os.path.exists(path):
        os.makedirs(path)
```

03
爬取选手图片数据

创建存放每个选手图片的文件夹：首先生成选手姓名文件夹；如果该路径下不存在指定文件夹，就使用os.makedirs()函数依次创建指定名称的文件夹。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

```
for i, pic_url in enumerate(pic_urls):
    try:
        pic = requests.get(pic_url, timeout=15)
        string = str(i + 1) + '.jpg'
        with open(path+string, 'wb') as f:
            f.write(pic.content)
            #print('成功下载第%s张图片: %s'% (str(i+1), str(pic_url)))
```

03

爬取选手图片数据

爬取选手图片数据：遍历图片链接列表，根据每一张目标图片的链接网址，使用requests.get()函数发送请求，获取指定图片的链接页面；打开指定路径下已创建的jpg文件，写入图片链接页面上的内容，即该选手的图片数据。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

```
        except Exception as e:
            #print('下载第%s张图片时失败: %s'%
(str(i + 1), str(pic_url)))
            print(e)
            continue
```

03
爬取选手图片数据

进行异常处理：若图片信息读取有误，则输出相应的异常信息，并跳出本次循环，继续爬取下一个图片链接页面的内容。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

04

爬取每位参赛选手的百度百科个人页面信息

根据函数parse_wiki_data()生成的stars.json文件，读取其中每位选手的百度百科个人页面网址，爬取对应的二级页面信息并保存。因此，自定义选手个人页面信息爬取函数crawl_everyone_wiki_urls()，主要源代码如下：

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

```
def crawl_everyone_wiki_urls():
    with open('work/' + 'stars.json', 'r', encoding='UTF-8') as file:
        json_array = json.loads(file.read())
    headers = {'User-
Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KH
TML, like Gecko) Chrome/67.0.3396.99 Safari/537.36'}
    star_infos = []
    for star in json_array:
        star_info = {}
        name = star['name']
        link = star['link']
        star_info['name'] = name
```

04

爬取每位参赛选手的百度百科个人页面信息

定义请求头部，读取stars.json文件中所有内容的字符串，并转化为字典元素组成的列表变量json_array，遍历列表对象json_array中的每个元素，获取选手姓名和个人页面信息链接网址，分别保存在字典变量star_info和字符串变量link中。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

```
  #获取选手的民族、星座、血型、体重等信息
base_info_div = bs.find('div',{'class':'basic-info cmn-clearfix'})
dls = base_info_div.find_all('dl')
for dl in dls:
    dts = dl.find_all('dt')
    for dt in dts:
        if "".join(str(dt.text).split()) == '民族':
            star_info['nation'] = dt.find_next('dd').text
        if "".join(str(dt.text).split()) == '星座':
            star_info['constellation'] = dt.find_next('dd').text
        if "".join(str(dt.text).split()) == '血型':
            star_info['blood_type'] = dt.find_next('dd').text
        if "".join(str(dt.text).split()) == '身高':
            height_str = str(dt.find_next('dd').text)
            star_info['height'] = str(height_str[0:height_str.rfind('cm')]).replace("\n","")
        if "".join(str(dt.text).split()) == '体重':
            star_info['weight'] = str(dt.find_next('dd').text).replace("\n","")
        if "".join(str(dt.text).split()) == '出生日期':
            birth_day_str = str(dt.find_next('dd').text).replace("\n","")
        if '年' in  birth_day_str:
            star_info['birth_day'] = birth_day_str[0:birth_day_str.rfind('年')]
star_infos.append(star_info)
```

获取选手的个人文本信息：依据字典类型的标签属性名及属性值，查找“div”标签第一次出现的位置并存放于变量base_info_div中，找到其子节点中的所有“dl”标签、孙节点中的所有“dt”标签，进而找到该标签下的民族、星座、血型、体重等标签文本，获取选手的个人文本信息并保存至字典变量star_info，追加至选手个人信息列表变量star_infos。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

使用print函数输出base_info_div变量的部分内容，示例如下：

```
out[]:<div class=" basic-info cmn-clearfix" >
      <dl class=" basicInfo-block basicInfo-left" >
      <dt class=" basicInfo-item name" >中文名</dt>
      <dd class=" basicInfo-item value" >
      符莹
      </dd>
      <dt class=" basicInfo-item name" >别　名</dt>
      <dd class=" basicInfo-item value" >
      阿朵
      </dd>
      … … … …
      <dt class=" basicInfo-item name" >星　座</dt>
      <dd class=" basicInfo-item value" >
      <a data-lemmaid=" 23547" href=" /item/%E7%99%BD%E7%BE
  %8A%E5%BA%A7/23547" target=" _blank" >白羊座</a>
      </dd>
      … … … …
```

04

爬取每位参赛选手的百度百科个人页面信息

从输出结果可以看到，每个“dt”标签下的别名、星座、血型、体重等标签文本中混杂着空格，因此代码22~40行在获取每个标签文本时使用“str(dt.text).split()”方法切分每个标签文本dt.text生成的字符串，通过“"".join()”方法去掉标签文本中无用的空格，从而匹配到“别名”、“星座”等文字，生成字典变量star_info中对应的键值对元素，其中必要时还要去掉可能的回车换行符等。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

04

爬取每位参赛选手的百度百科个人页面信息

```
#从个人百度百科页面中解析得到一个链接，该链接指向选手图片列表页面
if bs.select('.summary-pic a'):
    pic_list_url = bs.select('.summary-pic a')[0].get('href')
    pic_list_url = 'https://baike.baidu.com' + pic_list_url
```

使用bs.select()方法获得页面中“a”标签下“href”属性中选手图片列表页面的相对地址数据，进而生成选手图片列表页面绝对地址并存放在变量pic_list_url中。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

使用print函数输出“bs.select('.summary-pic a')[0]”对象的部分内容，示例如下：

04
爬取每位参赛选手的百度百科个人页面信息

```
out[]:<a href="/pic/%E9%98%BF%E6%9C%B5/435383/1/aa18972bd40735fa08a19ee091510fb30f240812?fr=lemma&amp;ct=single" nslog-type="10002401" target="_blank">
        <img src="https://bkimg.cdn.bcebos.com/pic/aa18972bd40735fa08a19ee091510fb30f240812?x-bce-process=image/resize,m_lfit,w_268,limit_1/format,f_jpg"/>
        <button class="picAlbumBtn"><em></em><span>图集</span></button>
        <div>阿朵的概述图（1张）</div>
        </a>
```

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

```
#向选手图片列表页面发送http get请求
pic_list_response = requests.get(pic_list_url,headers=headers)
#对选手图片列表页面进行解析，获取所有图片链接
bs = BeautifulSoup(pic_list_response.text,'lxml')
pic_list_html=bs.select('.pic-list img ')
pic_urls = []
for pic_html in pic_list_html:
    pic_url = pic_html.get('src')
    pic_urls.append(pic_url)
#根据图片链接列表pic_urls, 下载所有图片，保存在以name命名的文件夹中
down_save_pic(name,pic_urls)
```

04
爬取每位参赛选手的百度百科个人页面信息

保存选手个人信息：将stars变量转换为字符串类型，其中可能的“\'”符号统一为“\"”符号，去掉网页中可能混杂的空白字符“\xa0”，然后通过json.loads()函数转换为Python列表数据类型，最后写入work目录下的stars_info.json文件。至此，函数完成了参赛选手百度百科个人页面数据解析。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

04

爬取每位参赛选手的百度百科个人页面信息

```
    #将个人信息存储到json文件中
json_data = json.loads(str(star_infos).replace("\'","\"").replace("\\xa0",""))
with open('work/' + 'stars_info.json', 'w', encoding='UTF-8') as f:
    json.dump(json_data, f, ensure_ascii=False)
```

从输出结果可以看到，每个“dt”标签下的别名、星座、血型、体重等标签文本中混杂着空格，因此代码22~40行在获取每个标签文本时使用“str(dt.text).split()”方法切分每个标签文本dt.text生成的字符串，通过“"".join()”方法去掉标签文本中无用的空格，从而匹配到“别名”、“星座”等文字，生成字典变量star_info中对应的键值对元素，其中必要时还要去掉可能的回车换行符等。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

数据获取

在主程序中，依次调用上述四个自定义函数，完成综艺节目《乘风破浪的姐姐》参赛选手数据爬取任务。主要代码如下：

05
数据爬取主程序

```
1  if __name__ == '__main__':
2      #爬取百度百科中《乘风破浪的姐姐》中参赛选手信息，
3  返回 html
4      html = crawl_wiki_data()
5      #解析 html,得到选手信息，保存为 json 文件
6      parse_wiki_data(html)
7      #从每个选手的百度百科页面上爬取,并保存
8      crawl_everyone_wiki_urls()
       print("所有信息爬取完成！")
```

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

可视化数据分析

本书第七章《平凡的荣耀》数据分析案例中，采用扩展库numpy读写json文件中的数据，操作步骤相对繁琐。本案例使用数据分析常用扩展库pandas读写json数据文件，重点讲解基于pandas和matplotlib的可视化数据分析代码实现。AI Studio平台中同时给出了两种方式的源代码，便于读者进行代码分析、比较和验证。

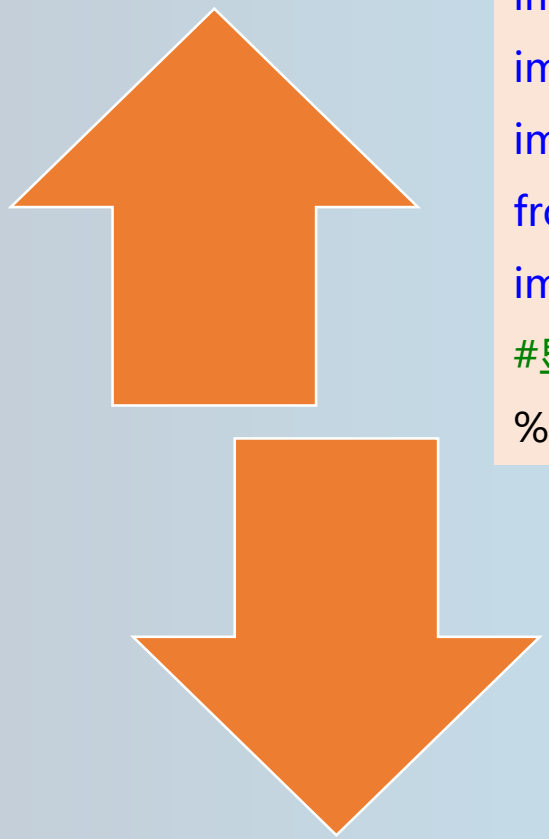
综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

可视化数据分析

01

绘制参赛选手年龄分布柱状图



```
import numpy as np
import json
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
import pandas as pd
#显示matplotlib生成的图形
%matplotlib inline
```

准备工作：为读取json文件导入json标准库；为显示中文标签、标题等导入matplotlib及其子模块matplotlib.font_manager；为使用pandas库函数进行数据分析导入pandas及其基础库numpy；为数据可视化导入matplotlib及其子模块matplotlib.pyplot；为正确显示生成的图形加入魔术命令%matplotlib inline。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

可视化数据分析

01

绘制参赛选手年龄分布柱状图

```
df = pd.read_json('work/stars_info.json',dtype = {'birth_day' : str})
df = df[df['birth_day'].map(len) == 4]
grouped=df['name'].groupby(df['birth_day'])
s = grouped.count()
birth_days_list = s.index
count_list = s.values
```

生成绘图数据：利用扩展库pandas中的pd.read_json()函数读取json文件，参数为要读取的文件路径“work/stars_info.json”，返回列、行形式存储的Pandas DataFrame数据，其中“birth_day”列为字符串类型的选手出生年份数据，部分内容示例如下页所示：

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

01

绘制参赛选手年龄分布柱状图

```
out[]: name   nation     birth_day constellation    blood_type  \
0  阿朵  \n土家族\n  1980       \n白羊座\n     \nO型\n
1  白冰   \n汉族\n   1986       \n金牛座\n     \nAB型\n
2  陈松伶  \n汉\n    1971       \n水瓶座\n       NaN
3  丁当   \n汉族\n   1982       \n白羊座\n     \nO型\n
4  黄圣依  \n汉族\n  1983       \n水瓶座\n     \nA型\n
5  黄龄   \n汉族\n   1987       \n水瓶座\n       NaN
6  海陆   \n汉族\n   1984       \n天秤座\n     \nO型\n
7  金晨   \n回族\n   1990       \n处女座\n       NaN
8  金莎   \n汉族\n  一说1981年，一说1983       \n双鱼座\n     \nA型\n
9  蓝盈莹  \n畲族\n  1990       \n白羊座\n     \nO型\n
10 李斯丹妮  \n回族\n  1990     \n金牛座\n       NaN
11 刘芸   \n汉族\n   1982       \n摩羯座\n     \nB型\n
12 孟佳 \n汉族\n  1989年2月3日(微博认证信息显示为1990 \n水瓶座\n \nO型\n
13宁静   \n汉族\n   1972       \n金牛座\n     \nAB型\n
```

从输出结果可以看到，行索引为8和12的选手出生年份不确定，代码第10行用于去掉出生年份长度不等于4的行数据，完成出生年份数据清洗。然后使用groupby()方法对经过数据预处理的DataFrame索引列“name”按照出生年份进行分组、聚合操作，统计属于不同出生年份的选手数目并保存在Series数组变量“s”中，其中列索引名是出生年份，保存为变量“birth_days_list”，对应每个出生年份的统计数据保存为“count_list”变量。

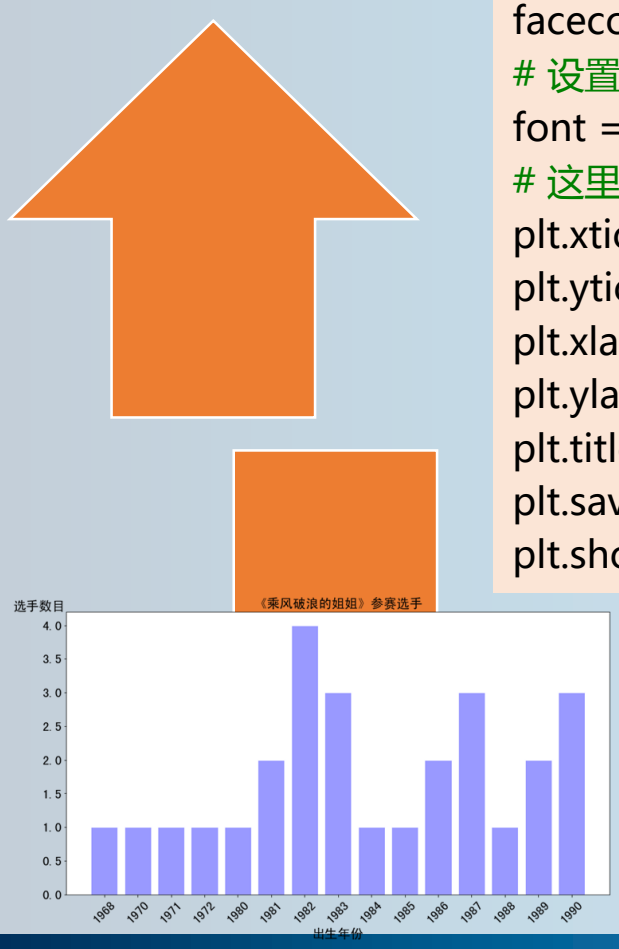
综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

可视化数据分析

01

绘制参赛选手年龄分布柱状图




```
plt.figure(figsize=(15,8))
plt.bar(range(len(count_list)), count_list, color='r', tick_label=birth_days_list,
facecolor='#9999ff',edgecolor='white')
# 设置显示中文
font = FontProperties(fname="/home/aistudio/work/simhei.ttf", size=20)
# 这里是调节横坐标的倾斜度，rotation是度数，以及设置刻度字体大小
plt.xticks(rotation=45,fontsize=20,fontproperties=font)
plt.yticks(fontsize=20,fontproperties=font)
plt.xlabel('出生年份',fontproperties=font)
plt.ylabel('选手数目',rotation='horizontal',y=1,fontproperties= font)
plt.title('《乘风破浪的姐姐》参赛选手', fontsize = 24, fontproperties=font )
plt.savefig('/home/aistudio/work/bar_result02.jpg')
plt.show()
```

绘制柱状图：定义画布尺寸后，依据出生年份统计结果绘制柱状图，x轴为不同出生年份的选手数据，x轴刻度标签为出生年份变量“birth_days_list”，y轴为出生年份统计结果；x轴刻度标签旋转45度，显示为中文文本；显示坐标轴标签、中文图例和中文标题；生成的图形保存在指定文件夹下的jpg文件中；

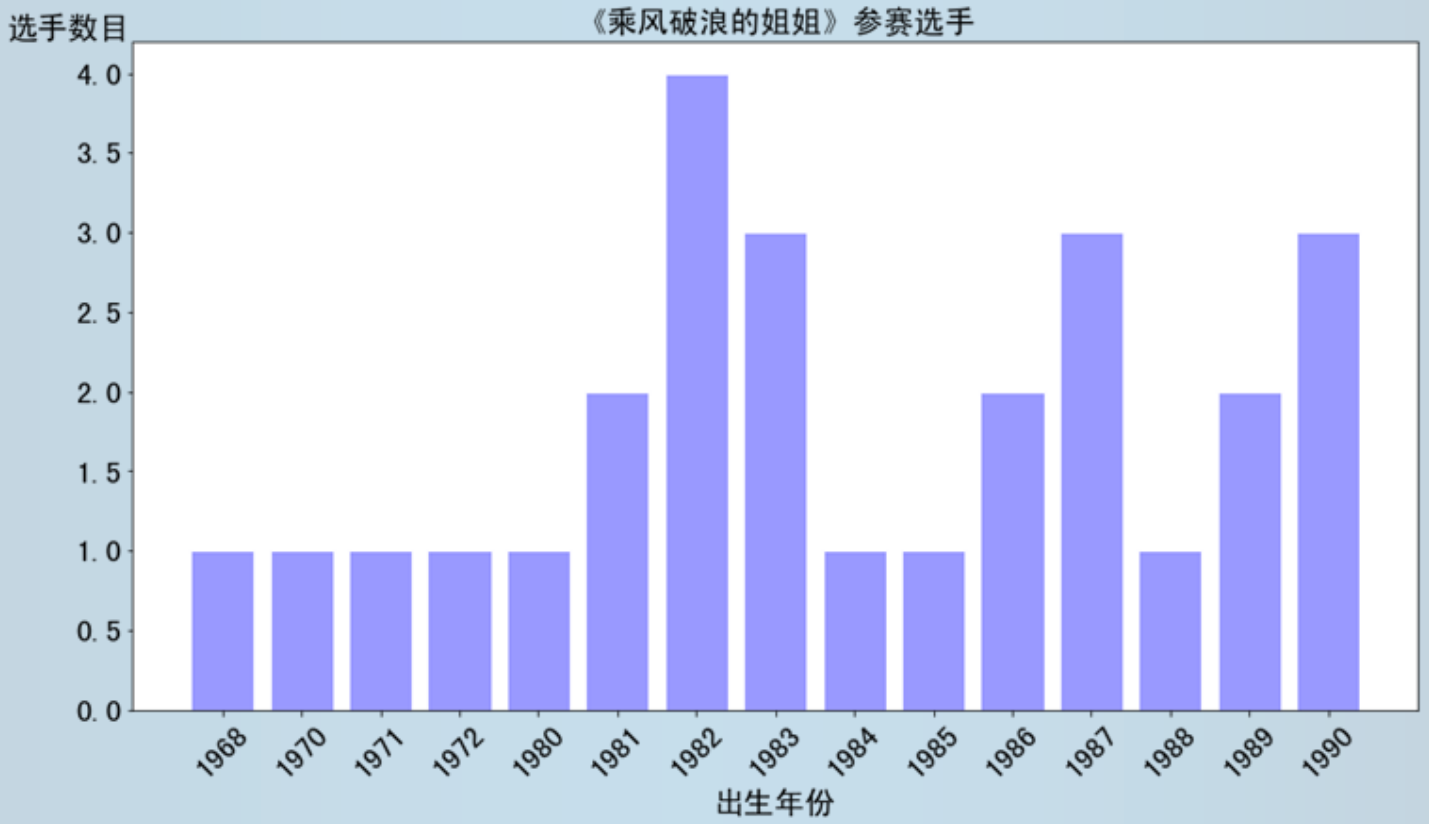
综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

可视化数据分析

01

绘制参赛选手年龄分布柱状图



综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

可视化数据分析

02

绘制选手体重饼状图

代码9-11行

```
df = pd.read_json('work/stars_info.json')
weights=df['weight']
arrs = weights.values
```

读取数据：读取json文件，返回Pandas DataFrame二维数组，获取索引列“weight”对应的Series数据值，保存至ndarray数组变量arrs。使用print函数输出变量arrs中的数据，如下所示。

out[]: [nan '50 kg' nan '48 kg' '45 kg' '46 kg' '48 kg' '43 kg[15]' '46 kg' '49 kg' '53 kg' '49 kg' '45 kg' '107 斤' nan '46 kg' '46 kg' '47 kg' '48 kg' '44 kg' nan '45 kg' '51 kg' '48 kg' nan '50 kg' '49 kg' '49 kg' nan '47 kg']

显然，变量arrs中包含着空值“nan”、不规范数据“43 kg[15]”和格式不统一的数据“107 斤”。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

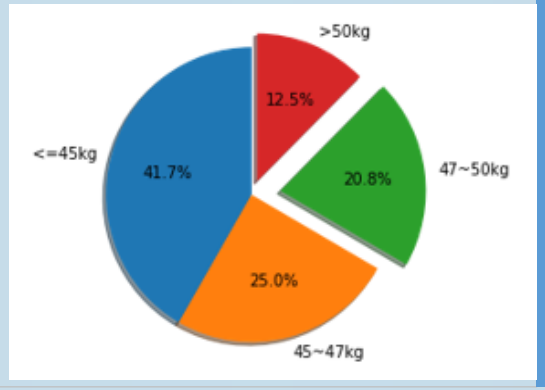
可视化数据分析

02

绘制选手体重饼状图

代码13-16行

```
arrs = [x for x in arrs if not pd.isnull(x)]
for i in range(len(arrs)):
    arrs[i] = float(arrs[i][0:2])
arrs[10]=53.5
```



数据预处理：代码第13行可以去除空值；代码14~15行提取选手体重前两位数据并转换为float类型，其中arrs[10]元素的值“107 斤”出现数据转换错误；代码16行单独处理arrs[10]元素，取值设置为该选手体重的公斤数。

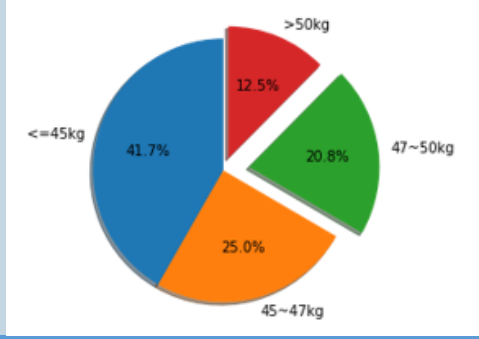
综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

可视化数据分析

02

绘制选手体重饼状图



代码18-22行

```
#pandas.cut用来把一组数据分割成离散的区间。
bin=[0,45,47,50,55]
se1=pd.cut(arrs,bin)
#pandas.value_counts()可以对Series里面的每个值进行计数并且排序。
sizes = pd.value_counts(se1)
```

指定饼图中各扇形的取值区间，使用pandas.cut()函数把一组数据分割成离散的区间。这里将一组体重数据，分割成不同的体重区间，其中参数bins表示分割后的区间范围，返回一个“pandas.core.arrays.categorical.Categorical”类型的数据，存放于变量se1，代表划分区间后体重数据中的每个值在哪个区间。

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

02

绘制选手体重饼状图

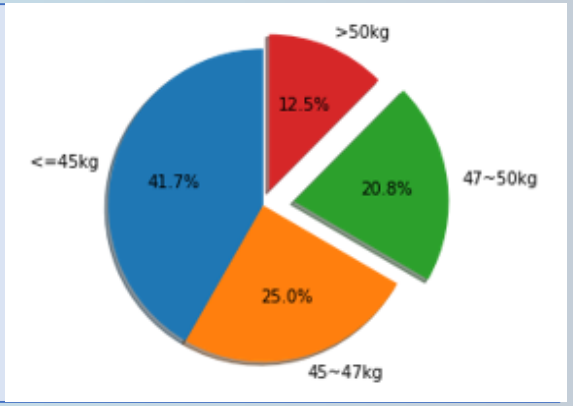
可视化数据分析

使用print函数输出变量se1的内容：

```
out[]:[(47, 50], (47, 50], (0, 45], (45, 47], (47, 50], ..., (47, 50], (47, 50], (47, 50], (47, 50], (45, 47]]
      Length: 24
      Categories (4, interval[int64]): [(0, 45] < (45, 47] < (47, 50] < (50, 55]]
```

接着使用pandas.value_counts()函数对变量se1的每个值进行计数并且排序，返回s类型的数组数据，存放于变量sizes，使用print函数输出变量sizes的内容：

```
out[]:(47, 50]    10
      (45, 47]     6
      (0, 45]      5
      (50, 55]     3
      dtype: int64
```



综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

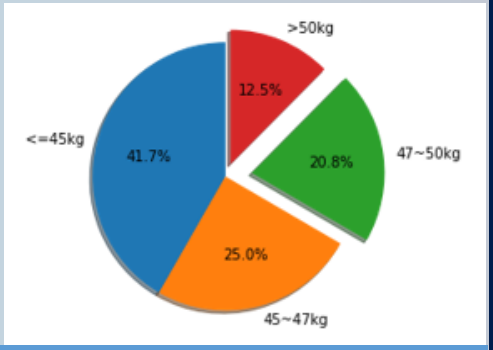
可视化数据分析

02

绘制选手体重饼状图

代码24-31行

```
labels = '<=45kg','45~47kg', '47~50kg', '>50kg'
explode = ( 0, 0, 0.2, 0.1)
fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')
plt.savefig('/home/aistudio/work/pie_result02.jpg')
plt.show()
```



绘制选手体重饼状图：使用ax1.pie()方法绘制饼图，其中变量sizes为扇形区域的数据序列；元组变量explode表示每个扇形区域偏离中心的距离；元组变量labels表示每个扇形对应的标签内容；参数autopct设置扇形内显示的标签内容，表示扇形区域内的标签数据为百分数，取值保留至小数点后一位；参数shadow设置饼图显示阴影；参数startangle设置第一个扇形的起始角度为90度，绘制圆形饼图。最后，保存绘图结果并显示.

综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

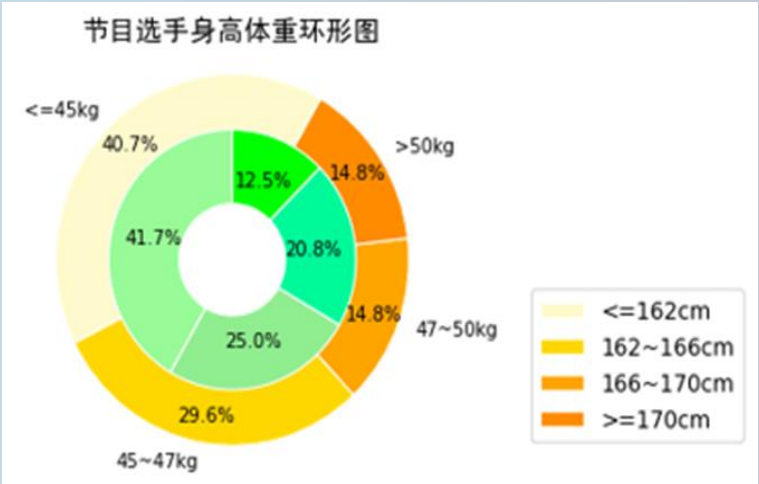
可视化数据分析

03

绘制参赛选手身高体重环形图

```
df2 = pd.read_json('work/stars_info.json')
heights=df2['height']
arrs2 = heights.values
arrs2 = [x for x in arrs2 if not pd.isnull(x)]
#pandas.cut把一组数据分割成离散的区间
bin=[0,162,166,170,175]
se2=pd.cut(arrs2,bin)
#pandas的value_counts()函数可以对Series里面的每个值进行计数并且排序。
pd.value_counts(se2)
#设置中文默认字体
font = FontProperties(fname="/home/aistudio/work/simhei.ttf", size=15)
labels2 = '<=162cm','162~166cm','166~170cm', '>=170cm'
sizes2 = pd.value_counts(se2)
```

使用类似源代码获取选手身高区间统计数据，存放于变量sizes2；



综艺节目选手数据爬取与探索性分析

Data crawling and exploratory analysis of variety show players

可视化数据分析

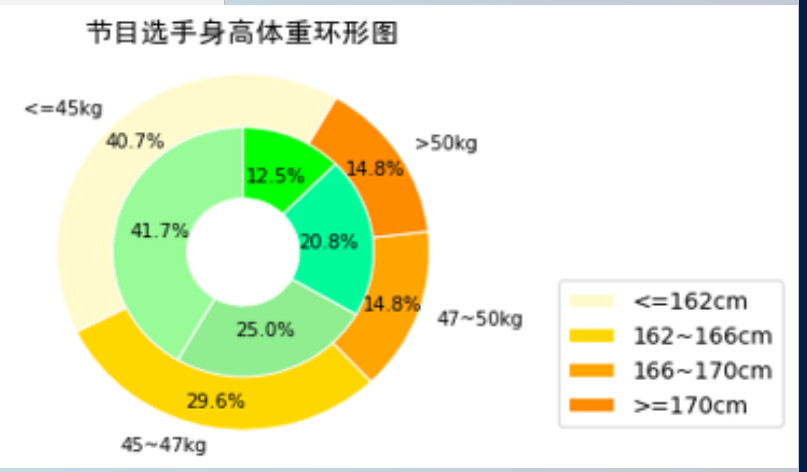
03

绘制参赛选手身高体重环形图

```
fig1, ax1 = plt.subplots()
color1=('lemonchiffon','gold','orange','darkorange')
color2=('palegreen','lightgreen','mediumspringgreen','lime','limegreen')

ax1.pie(sizes2, labels=labels, autopct='%1.1f%%',radius=1.0,textprops={'color':'black'},colors
=color1,pctdistance=0.85,startangle=60,wedgeprops={'width':0.4,'edgecolor':'w'})
ax1.pie(sizes,  autopct='%1.1f%%',radius=0.7,textprops={'color':'black'},colors=color2,pctdist
ance=0.65,startangle=90,wedgeprops={'width':0.4,'edgecolor':'w'})
ax1.axis('equal')
plt.legend((loc="lower right", labels=labels2, fontsize=12,
bbox_to_anchor=(1.35,0.05),borderaxespad=0.2)
plt.title("节目选手身高体重环形图",fontsize = 24,fontproperties=font)
plt.savefig('/home/aistudio/work/pie_result03.jpg')
plt.show()
```

绘制参赛选手身高体重环形图，设置不同的颜色，环形半径等参数，分别绘制两个不同色系的环形



PART TWO

波士顿房价预测

The prediction of house price in Boston

波士顿房价预测

The prediction of house price in Boston

任务描述

波士顿房价数据集包含506个样本，分为404个训练样本和102个测试样本。每个样本包含房屋以及房屋周围的详细信息，例如城镇犯罪率，一氧化氮浓度，住宅平均房间数，到中心区域的加权距离以及自住房平均房价等等。

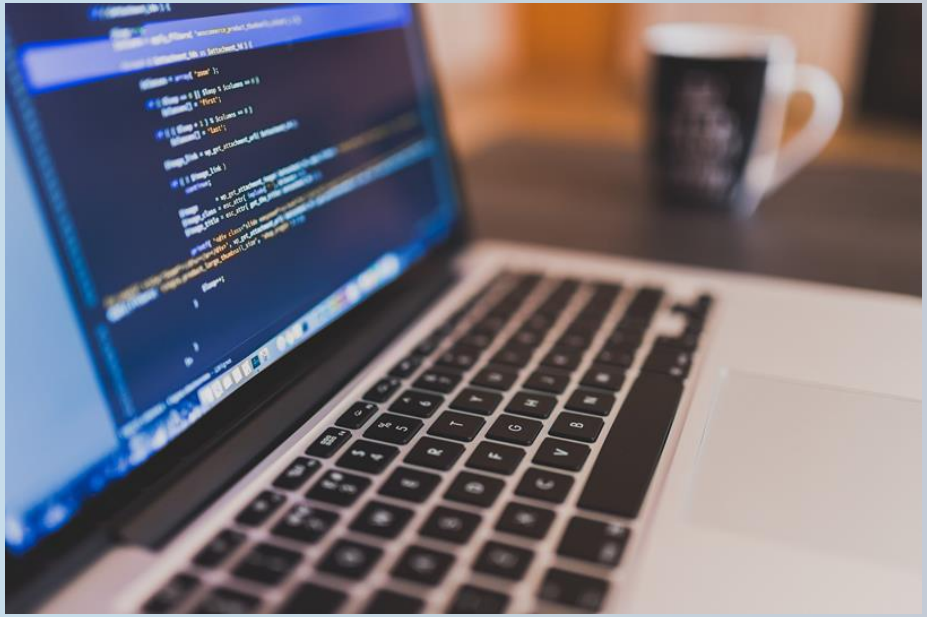
本案例首先依据波士顿房价数据集的13个输入变量和输出1个变量，进行探索性数据分析，通过数据可视化方法探索数据结构和数据间的规律；然后利用Python扩展库sklearn提供的经典机器学习模型预测当时该地区房屋价格的中位数，这是一个回归问题；接着构建神经网络模型训练和测试波士顿房价数据集，并进行模型评价。

波士顿房价预测

The prediction of house price in Boston

数据集介绍

在探索性数据分析阶段，本案例从百度飞桨AI Studio平台（网址：https://aistudio.baidu.com/aistudio/datasetoverview）下载保存波士顿房价数据集的文件“housing.data”。该文件存储了506条美国波士顿房价数据，每条数据前13列为数值型数据，分别描述了指定房屋的13项特征，最后一列为目标房价。为了便于数据读取和处理，首先将数据文件“housing.data”另存为“boston_housing.csv”，保存在AI Studio平台work文件夹下。



波士顿房价预测

The prediction of house price in Boston

数据集介绍

在验证性数据分析阶段，本案例借助数据加载器，使用sklearn模块下datasets子模块直接加载常用的波士顿房价数据集。

```
In[ ]:from sklearn.datasets import load_boston
      boston = load_boston()
      print(boston.DESCR) # 输出数据描述
```

```
out[ ]: ... _boston_dataset: Boston house prices dataset
**Data Set Characteristics:**
   :Number of Instances: 506
   :Number of Attributes: 13 numeric/categorical predictive.
Median Value (attribute 14) is usually the target.
   :Attribute Information (in order):
       - CRIM     per capita crime rate by town
       - ZN       proportion of residential land zoned for lots
over 25,000 sq.ft
... ... ... ...
:Missing Attribute Values: None
   :Creator: Harrison, D. and Rubinfeld, D.L.
... ... ... ...
```

波士顿房价预测

The prediction of house price in Boston

数据集介绍

通过DESCR属性查看数据集文档得知，该数据集中没有缺失的属性或特征值，波士顿房价属性描述如表所示。

变量名	属性描述	说明
CRIM	per capita crime rate by town	城镇人均犯罪率
ZN	proportion of residential land zoned for lots over 25,000 sq.ft.	超过25000平方英尺的住宅用地所占比例
INDUS	proportion of non-retail business acres per town	城镇非商业用地所占比例
CHAS	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)	查尔斯河虚拟变量(如果土地在河边=1；否则为0)
NOX	nitric oxides concentration (parts per 10 million)	环保指数：一氧化氮浓度(每1000万份)
RM	average number of rooms per dwelling	平均每栋住宅的房间数
AGE	proportion of owner-occupied units built prior to 1940	1940年之前建成的所有者自住单位的比例
DIS	weighted distances to five Boston employment centres	与五个波士顿就业中心的加权距离
RAD	index of accessibility to radial highways	距离高速公路的可达性指数
TAX	full-value property-tax rate per $10,000	每10,000美元的全额物业税率
RTRATIO	pupil-teacher ratio by town	城镇师生比例
B	1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town	1000(Bk-0.63)^2其中Bk是城镇黑人的比例
LSTAT	% lower status of the population	较低收入阶层的房东数目所占百分数
MEDV	Median value of owner-occupied homes in $1000's	(目标变量/类别属性)以1000美元计算的自有房屋房价的中位数

波士顿房价预测

The prediction of house price in Boston

数据集介绍

```
In[ ]:import pandas as pd
      train = pd.DataFrame(boston.data)
      train.columns=['crim','zn','indus','chas','nox','rm','age','dis','rad','tax','ptratio','b','lstat']
      train.columns=['人均犯罪','用地','非商业地','河','环保指标','房间数','老房子比例','加权距离','交通便利','税率','教师学生比','黑人比','低收入房东比' ]
      train
```

	人均犯罪	用地	非商业地	河	环保指标	房间数	老房子比例	加权距离	交通便利	税率	教师学生比	黑人比	低收入房东比
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...

波士顿房价预测

The prediction of house price in Boston

数据集介绍

输入数据是二维数组，每行对应一套房屋数据，每套房屋数据包括13个特征，每个特征有不同的取值范围：有的特征取值是比例，范围[0,1]；有的特征取值范围是1~12；还有的特征取值范围是[0,100]。

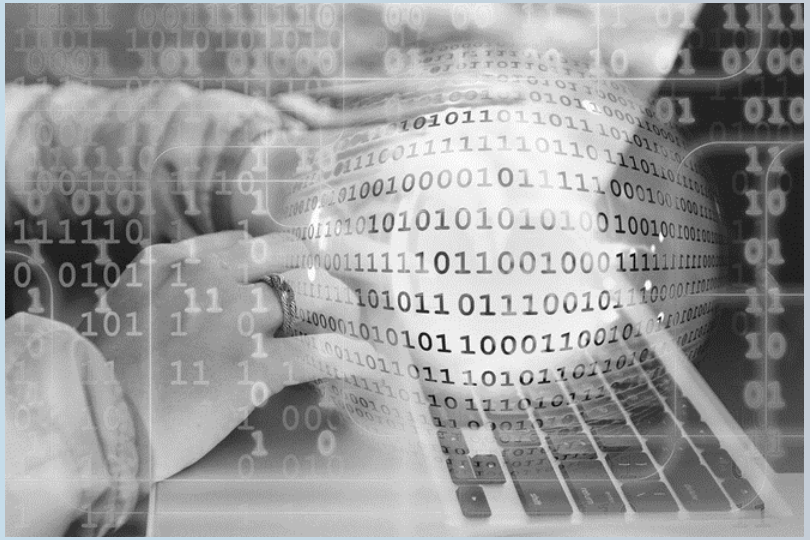
	人均犯罪	用地	非商业地	河	环保指标	房间数	老房子比例	加权距离	交通便利	税率	教师学生比	黑人比	低收入房东比
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...

波士顿房价预测

The prediction of house price in Boston

数据探索

通过Python命令可以简单地观察数据及其描述，查看数据间的关系，但不能直观显示数据内部的联系。下面本案例利用扩展库matplotlib进行数据可视化，进一步理解波士顿房价数据集的特点。



波士顿房价预测

The prediction of house price in Boston

数据探索

01
描述性数据统计

使用扩展库pandas的describe()方法可以展示数据的一些描述性统计信息，包括数据量、平均值、方差、中位数、四分位数和最值等，让用户对数据集有个初步的了解。

波士顿房价预测

The prediction of house price in Boston

数据探索

```
In[ ]:from pandas import read_csv
      import numpy as np
      import  json
      import matplotlib.pyplot as plt
      filename = 'work/boston_housing.csv'
      names = ['CRIM','ZN','INDUS','CHAS','NOX','RM','AGE','DIS','RAD','TAX','PTRATIO','LSTAT','MEDV']
      datas = read_csv(filename,names=names)
In[ ]:datas.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	452.000000
mean	13.295257	9.205158	0.140765	1.101175	15.679800	58.744660	6.173308	78.063241	339.317787	42.614980	332.791107	11.537806	23.750442
std	23.048697	7.169630	0.312765	1.646991	27.220206	33.104049	6.476435	203.542157	180.670077	87.585243	125.322456	6.064932	8.808602
min	0.000000	0.000000	0.000000	0.385000	3.561000	1.137000	1.129600	1.000000	20.200000	2.600000	0.320000	1.730000	6.300000
25%	0.000000	3.440000	0.000000	0.449000	5.961500	32.000000	2.430575	4.000000	254.000000	17.000000	364.995000	6.877500	18.500000
50%	0.000000	6.960000	0.000000	0.538000	6.322500	65.250000	3.925850	5.000000	307.000000	18.900000	390.660000	10.380000	21.950000
75%	18.100000	18.100000	0.000000	0.647000	6.949000	89.975000	6.332075	24.000000	403.000000	20.200000	395.615000	15.015000	26.600000
max	100.000000	27.740000	1.000000	7.313000	100.000000	100.000000	24.000000	666.000000	711.000000	396.900000	396.900000	34.410000	50.000000

代码调用pandas.read_csv()函数读取CSV数据文件“boston_housing.csv”，将返回的房价数据以pandas DataFrame二维数组的形式存储于变量datas，并通过列表变量names为该DataFrame数组指定字符串列索引。

波士顿房价预测

The prediction of house price in Boston

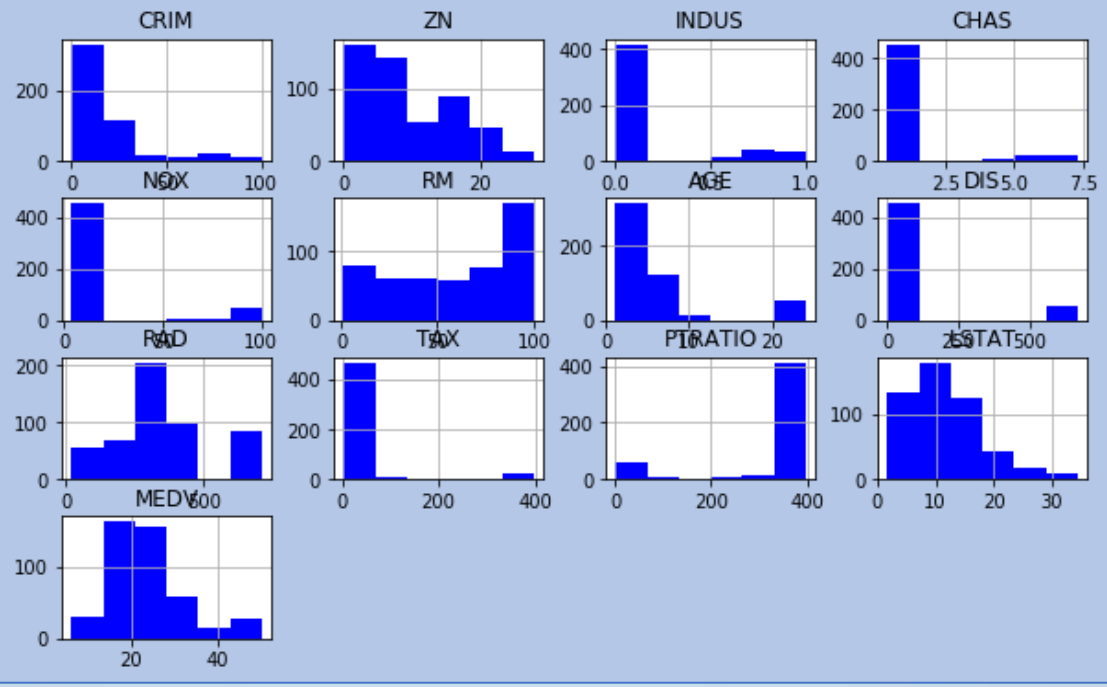
数据探索

02

直方图（质量分布图）

用一系列高度不等的纵横条纹来统计数据，横坐标表示数据类型，纵坐标表示数据分布情况。

```
In[ ]:datas.hist(figsize=(10,6),bins=6,color='b',density=False)
      plt.show()
```



波士顿房价预测

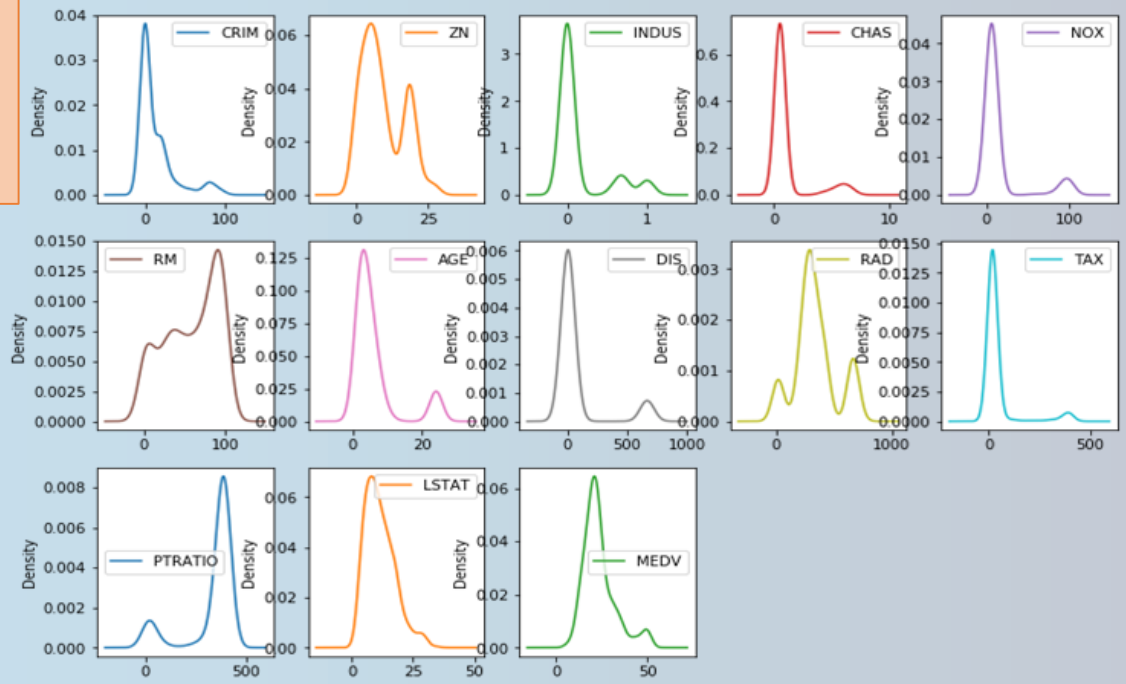
The prediction of house price in Boston

数据探索

取直方图的顶点数据，用平滑的曲线连接，生成基于直方图抽象得到的密度图，进一步查看数据的分布情况。

03
密度图

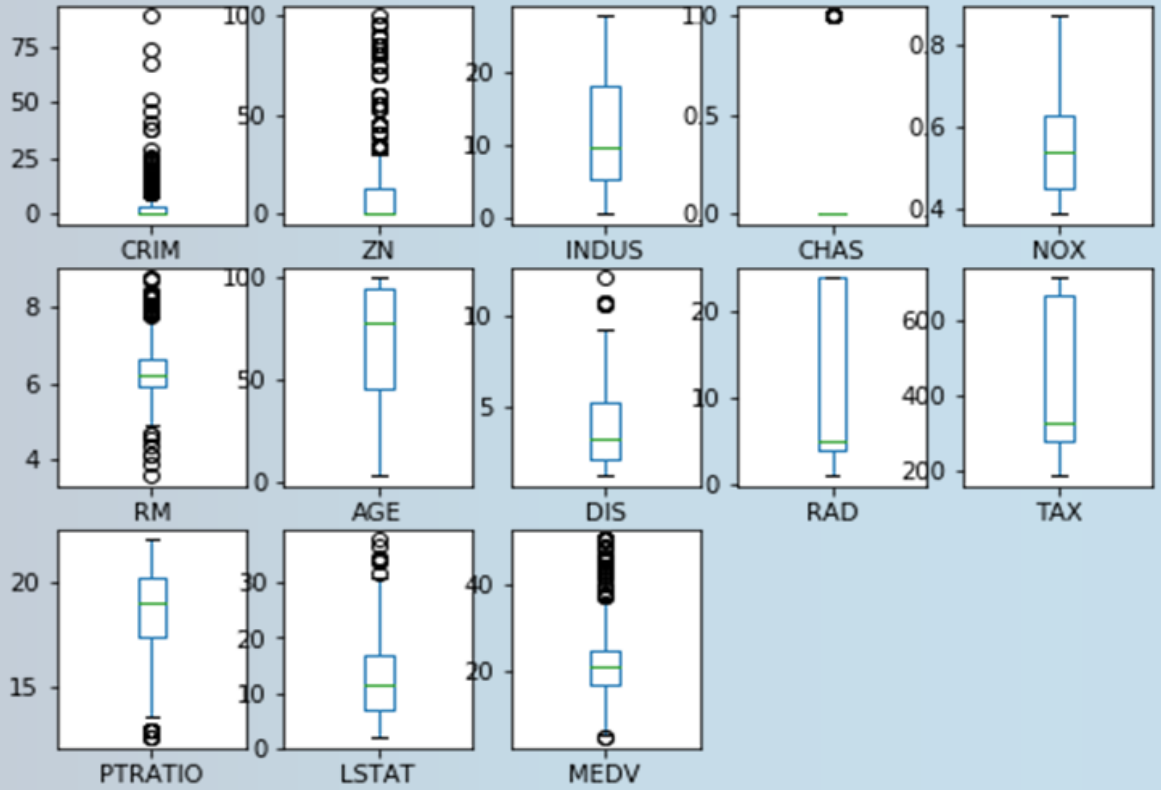
```
In[ ]:datas.plot(kind='density',subplots=True,
        layout=(3,5), sharex=False,figsize=(12,9))
        plt.show()
```



波士顿房价预测

The prediction of house price in Boston

数据探索



04

箱线图

箱线图中，基于中间的一条中位数（50%）线，然后分别是上下四分位数（75%、25%）线，接着是上、下各有一条边缘线。游离在上下边缘之外比中位数大1.5倍的是异常点。 通过箱线图可以查看数据的分散情况。

波士顿房价预测

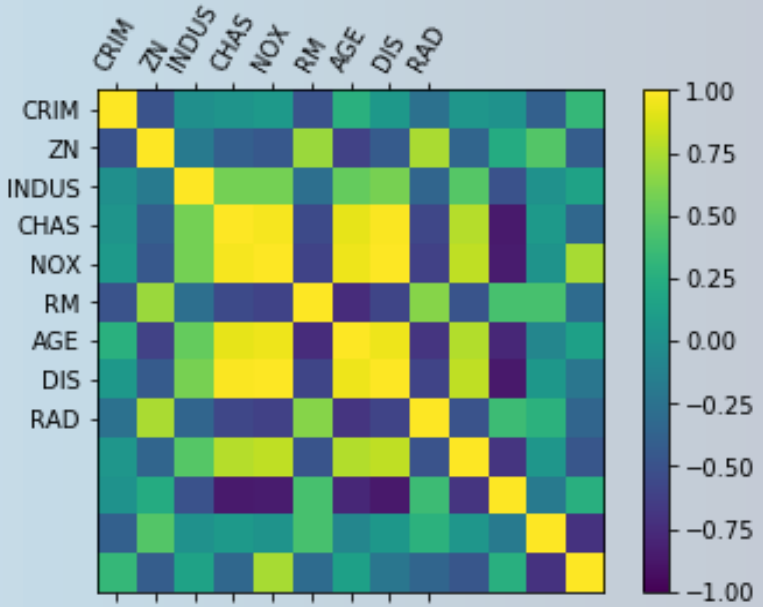
The prediction of house price in Boston

数据探索

矩阵图主要表示数据不同特征间相互影响的程度。如果两个特征的数据变化方向相同，那么它们正相关；若两个属性的数据变化方向相反，则呈现负相关关系。

05
矩阵图

```
In[ ]:corr = datas.corr()
      fig = plt.figure() #创建一个空图
      ax = fig.add_subplot(111)
      car = ax.matshow(corr,vmin=-1,vmax=1) #绘制热力图（矩阵图）
      fig.colorbar(car) #为matshow设置渐变色
      ticks = np.arange(0,9,1) #生成0-8
      ax.set_xticks(ticks) #生成刻度
      ax.set_yticks(ticks)
      ax.set_xticklabels(names)  #生成刻度标签
      ax.set_yticklabels(names，rotation=60)
```



波士顿房价预测

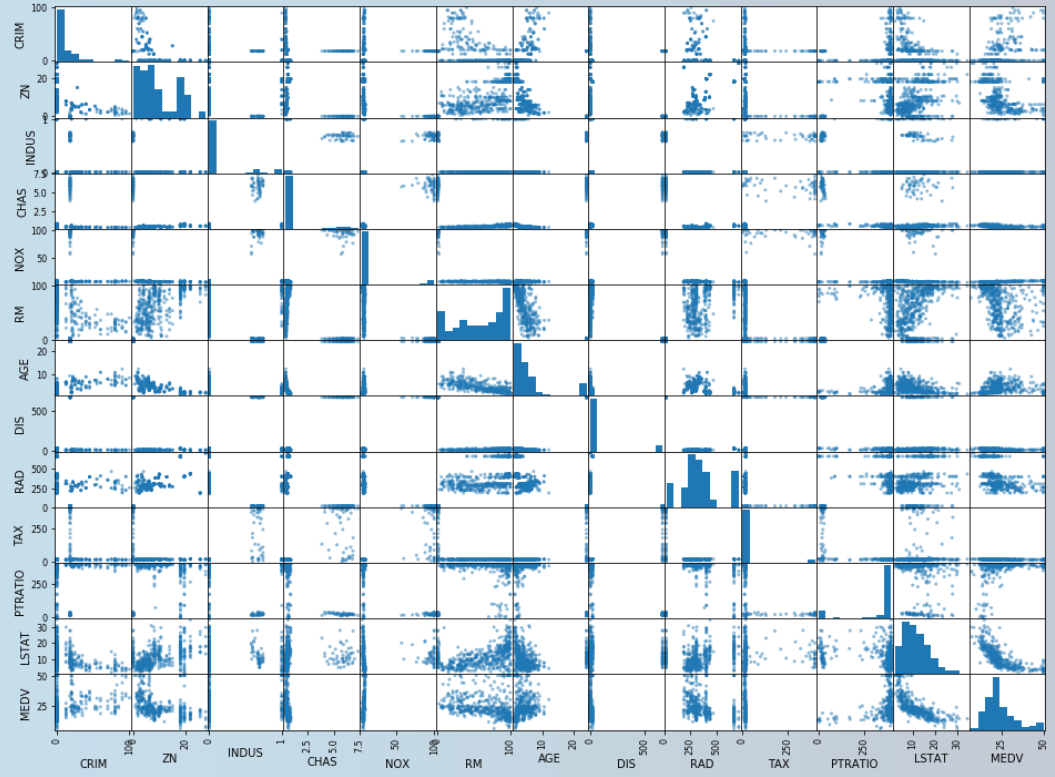
The prediction of house price in Boston

数据探索

06
散点矩阵

散点矩阵用于描述因变量随自变量而变化的大致趋势，能够形象地展示数据的线性相关性，可以依据散点矩阵的结果选择适合的数据点进行数据拟合。

```
In[ ]:from pandas.plotting import scatter_matrix
      scatter_matrix(datas,figsize=(16,12))#比corr展示的线性相关性更形象
      plt.show()
```



波士顿房价预测

The prediction of house price in Boston

数据预处理

01

Housing.data 文件内容的数据预处理

```
def load_data():
    datafile = 'work/housing.data' # 从文件导入数据
    data = np.fromfile(datafile, sep=' ')
    # 每条数据包括14项，其中前面13项是影响因素，第14项是房屋价格中位数
    feature_names = [ 'CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX'
, 'PTRATIO', 'B', 'LSTAT', 'MEDV']
    feature_num = len(feature_names)
     # 将原始数据reshape，成为[N, 14]的形状
    data = data.reshape([data.shape[0] // feature_num, feature_num])
```

实现数据变换：首先读取房价数据文件，使用np.fromfile()函数将文件内容转换为一维ndarray数组对象data，其中包含7084即506*14个元素；然后使用data.reshape()方法将一维数组转换成506行14列的二维数组，表示506行房价数据，每行包含14列数据，代码中data.shape[0]对象表示二维数组data的行数。

波士顿房价预测

The prediction of house price in Boston

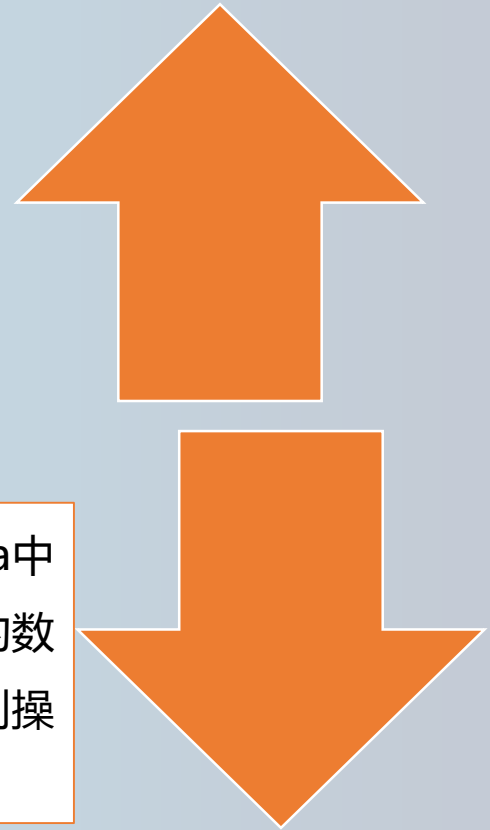
数据预处理

01

Housing.data 文件内容的数据预处理

```
# 将原数据集拆分成训练集和测试集，80%的数据做训练，20%的数据做测试
ratio = 0.8
offset = int(data.shape[0] * ratio)
training_data = data[:offset]
test_data = data[offset:]
# 计算训练集的最大值，最小值，平均值
maximums, minimums, avgs = training_data.max(axis=0), training_data.min(axis=0), training_data.sum(axis=0) / training_data.shape[0]
```

划分训练集和测试集：根据变量ratio取值0.8，将数据集对应的二维数组data中前80%的数据划分为训练集，存储在变量training_data中；将data中后20%的数据作为测试集，存储在变量test_data中；接着对训练集数据training_data按列操作，分别计算每个特征的最大值、最小值和平均值。



波士顿房价预测

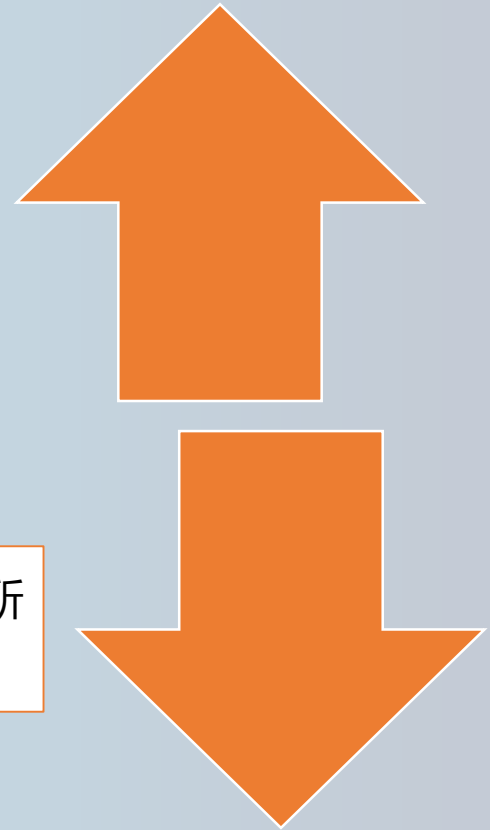
The prediction of house price in Boston

数据预处理

01

Housing.data 文件内容的数据预处理

```
# 对数据进行归一化处理
for i in range(feature_num):
    data[:, i] = (data[:, i]-avgs[i]) /(maximums[i]-minimums[i])
return training_data, test_data
```



完成数据归一化处理：根据公式6-2，采用min-max标准化方法对数据集中所有数据进行归一化处理；最后返回处理好的训练集数据和测试集数据。

波士顿房价预测

The prediction of house price in Boston

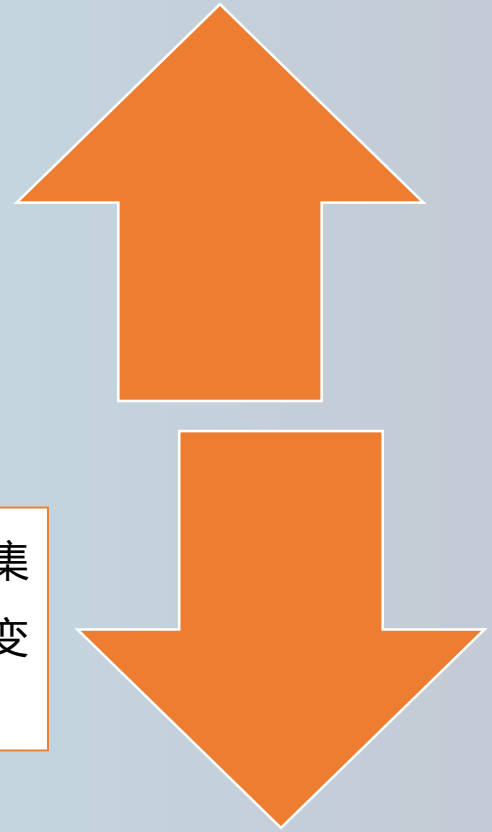
数据预处理

01

Housing.data 文件内容的数据预处理

```
#获取数据
training_data, test_data = load_data()
x = training_data[:, :-1]
y = training_data[:, -1:]
```

调用数据预处理函数：调用自定义函数load_data()完成数据预处理，将训练集中前13列数据即房价特征数据存入变量x；训练集最后一列数据即目标房价存入变量y。



波士顿房价预测

The prediction of house price in Boston

数据预处理

02

sklearn.datasets 模块加载的数据预处理

```
# step1 数据获取
from sklearn.datasets import load_boston
boston = load_boston()
# step2 数据分割
from sklearn.model_selection import train_test_split
import numpy as np
x = boston.data
y = boston.target
x_train,x_test,y_train,y_test = train_test_split(x, y, test_size = 0.25, random_state=33)
```

划分数据集：首先获取全部数据，将特征数据值和目标数据值分别保存在变量x和变量y中；然后使用扩展库sklearn子模块model_selection中的train_test_split函数对数据随机采样，将25%的数据用于测试，其余75%的数据用于模型训练，得到训练集特征值、测试集特征值、训练集目标值和测试集目标值，分别保存在变量x_train、x_test、y_train和y_test中。参数random_state设置随机数种子编号为33，为的是确保重复试验时得到相同的随机数以生成相同的数据。

波士顿房价预测

The prediction of house price in Boston

数据预处理

02

sklearn.datasets 模块加载的数据预处理

```
# 分析回归目标值的差异
print('The max target value is ', np.max(boston.target))
print('The min target value is ', np.min(boston.target))
print('The average target value is ', np.mean(boston.target))
# step3 数据标准化处理
from sklearn.preprocessing import StandardScaler
# 分别对特征和目标的标准化器进行初始化
ss_x = StandardScaler()
ss_y = StandardScaler()
x_train = ss_x.fit_transform(x_train)
x_test = ss_x.transform(x_test)
y_train = ss_y.fit_transform(y_train.reshape(-1,1)) #y_train转化为2维
y_test = ss_y.transform(y_test.reshape(-1,1))
```

对训练数据和测试数据进行标准化处理：首先使用StandardScaler()函数初始化两个标准化器，然后使用fit_transform()方法和transform()方法分别对训练数据和测试数据的特征值及目标值进行标准差标准化（standardScale）处理，即用数据值减去均值再除以方差。通过对每个特征维度去均值和方差归一化，使得处理后的数据符合均值为0，标准差为1的标准正态分布。其中fit_transform()方法包括fit和transform两个步骤，即先计算均值和标准差，然后进行数据转换，实现数据标准化；而transform()方法利用fit数值计算步骤得到的参数(μ，σ)，直接将数据转换成标准正态分布。使用fit_transform(x_train)方法和transform(x_text)方法，保证了对训练集和测试集的数据处理方式相同。

波士顿房价预测

The prediction of house price in Boston

数据预处理

02

sklearn.datasets 模块加载的数据预处理

数据探索结果显示目标房价的预测值之间差距很大，所以对房价的特征值和目标值进行了标准化处理。标准化处理之后的目标值发生了很大改变，用户可以使用inverse_transform()函数还原出真实结果，并且可以采用相同方法将预测的回归值还原。

波士顿房价预测

The prediction of house price in Boston

基于sklearn经典模型的房价预测

本案例采用扩展库sklearn中经典的机器学习模型，如线性回归、支持向量机（SVM）回归、K近邻回归、回归树和集成回归模型等对完成数据预处理的波士顿房价数据集进行训练和测试，完成房价预测任务。

波士顿房价预测

The prediction of house price in Boston

基于sklearn经典模型的房价预测

01
基于线性回归器的房价预测

代码第1和第6行：分别导入扩展库sklearn.linear_model中的线性回归器LinearRegression和SGDRegressor；

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression() # 使用默认参数初始化
lr.fit(x_train, y_train.ravel()) # y_train.ravel()将目标转化为1维
lr_y_pred = lr.predict(x_test)

from sklearn.linear_model import SGDRegressor
sgdr = SGDRegressor()
sgdr.fit(x_train, y_train.ravel())
sgdr_y_pred = sgdr.predict(x_test)
```

代码第3和第8行：使用训练数据分别进行LinearRegression和SGDRegressor模型的参数估计；

代码第2和第7行：使用默认参数分别创建线性回归模型LinearRegression和SGDRegressor；

代码第4和第9行：使用LinearRegression和SGDRegressor模型分别对测试数据进行回归预测。

波士顿房价预测

The prediction of house price in Boston

基于sklearn经典模型的房价预测

使用三种不同的核函数配置SVM，进行波士顿房价回归预测

```
from sklearn.svm import SVR
```

01

```
# 使用线性核函数进行回归预测
linear_svr = SVR(kernel='linear')
linear_svr.fit(x_train, y_train.ravel())
linear_svr_y_pred = linear_svr.predict(x_test)
```

02

```
# 使用多项式核函数进行回归预测
poly_svr = SVR(kernel='poly')
poly_svr.fit(x_train, y_train.ravel())
poly_svr_y_pred = poly_svr.predict(x_test)
```

02

基于支持向量机的房价预测

03

```
# 使用径向基函数进行回归预测
rbf_svr = SVR(kernel='rbf')
rbf_svr.fit(x_train, y_train.ravel())
rbf_svr_y_pred = rbf_svr.predict(x_test)
```

波士顿房价预测

The prediction of house price in Boston

基于sklearn经典模型的房价预测

02
基于支持向量机的房价预测

首先导入扩展库sklearn.svm中的支持向量机回归器SVR；然后创建支持向量机回归模型，通过设置参数kernel取值“linear”、“poly”和“rbf”，分别表示使用“线性核函数”、“多项式核函数”和“径向基函数”进行波士顿房价回归预测；接着分别对这三个模型进行训练，分析模型参数，用训练集数据拟合回归器模型；最后对三个模型进行测试，使用fit()方法计算得到参数并构建模型，对特征数据进行预测获得房价预测结果。

波士顿房价预测

The prediction of house price in Boston

基于sklearn经典模型的房价预测

03

基于K近邻回归器的房价预测

本书第六章“约会对象筛选”案例显示，K近邻分类模型不需要训练参数。在回归任务中，K近邻回归模型依据K个最接近训练样本特征的目标值，对待测样本的回归值进行预测。这是根据样本的相似程度预测回归值的方法。下面使用两种回归策略衡量待测样本的回归值。

波士顿房价预测

The prediction of house price in Boston

基于sklearn经典模型的房价预测

03

基于K近邻回归器的房价预测

```
from sklearn.neighbors import KNeighborsRegressor
```

01

```
# 方法1，预测方式为平均回归weights='uniform'
uni_knr = KNeighborsRegressor(weights='uniform')
uni_knr.fit(x_train, y_train)
uni_knr_y_pred = uni_knr.predict(x_test)
```

02

```
# 方法2，使用距离加权回归weights='distance'
dis_knr = KNeighborsRegressor(weights='distance')
dis_knr.fit(x_train, y_train)
dis_knr_y_pred = dis_knr.predict(x_test)
```

导入扩展库sklearn.neighbors中的K近邻回归器KNeighborsRegressor，初始化K近邻回归器后，首先调整配置，使用平均回归策略进行预测，然后采用加权平均的回归方法进行房价预测。

波士顿房价预测

The prediction of house price in Boston

基于sklearn经典模型的房价预测

04
基于回归树的房价预测

基于回归树进行房价预测时，回归树的叶子结点为连续型数据。依据训练数据得到回归树叶子节点的均值，进而决定最终的预测类别。

```
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor()
dtr.fit(x_train, y_train)
dtr_y_pred = dtr.predict(x_test)
```

导入扩展库sklearn.tree中的决策树回归器DecisionTreeRegressor，初始化回归器，用波士顿房价训练数据构建回归树，使用默认配置的单一回归树对测试数据进行预测，并将预测值存储在变量dtr_y_pred中。

波士顿房价预测

The prediction of house price in Boston

基于sklearn经典模型的房价预测

使用三种集成回归模型对波士顿房价数据进行回归预测。

分别导入扩展库sklearn.ensemble中的三种集成回归模型 RandomForestRegressor、ExtraTreesRegressor和GradientBoostingRegressor，对美国波士顿房价训练数据进行学习，并对测试数据进行房价预测。

05
基于集成回归模型的房价预测

01

```
from sklearn.ensemble import
RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(x_train, y_train.ravel())
rfr_y_pred = rfr.predict(x_test)
```

02

```
from sklearn.ensemble import
ExtraTreesRegressor
etr = ExtraTreesRegressor()
etr.fit(x_train, y_train.ravel())
etr_y_pred = etr.predict(x_test)
```

03

```
from sklearn.ensemble import
GradientBoostingRegressor
gbr = GradientBoostingRegressor()
gbr.fit(x_train, y_train.ravel())
gbr_y_pred = gbr.predict(x_test)
```

波士顿房价预测

The prediction of house price in Boston

利用Python构建网络模型进行房价预测

以“类和对象”的方式搭建网络结构并完成计算、预测和输出过程。首先基于Network类的定义设计模型的网络结构。

Step 1　网络模型构建

```
class Network(object):
    def __init__(self,num_of_weights):
        np.random.seed(0)
        self.w=np.random.randn(num_of_weights,1)
        self.b=0
```

__init__()函数进行初始化。为了保持每次运行程序时输出结果的一致性，这里设置固定的随机数种子。在Network类定义中，类成员变量包含参数w和b。随机产生w的初始值，偏移量b的初始值为0。

波士顿房价预测

The prediction of house price in Boston

利用Python构建网络模型进行房价预测

Step 1 网络模型构建		
	def forward(self,x): z=np.dot(x,self.w)+self.b return z	forward()函数实现模型的前向（从输入到输出）计算过程。将样本的特征向量x与参数向量w进行矩阵相乘，并与偏移量b相加，实现完整的线性回归公式，返回计算结果z作为预测值输出。
	def loss(self,z,y): error = z - y cost = error * error cost = np.mean(cost) return cost	loss()函数计算模型损失。使用模型计算出特征向量x影响下的房价预测值z，但实际上房屋特征数据对应的房价是y。需要衡量预测值z跟真实值y之间的差距error。对于回归问题，这里使用均方误差作为评价模型好坏的指标，即cost=(y−z)*(y−z)。考虑到每个样本的损失，因此对单个样本的损失函数求和除以样本总数，即求均值。

波士顿房价预测

The prediction of house price in Boston

利用Python构建网络模型进行房价预测

Step 1　网络模型构建

```
def gradient(self, x, y):
    z = self.forward(x)
    gradient_w = (z-y)*x
    gradient_w = np.mean(gradient_w, axis=0)
    gradient_w = gradient_w[:, np.newaxis]
    gradient_b = (z-y)
    gradient_b = np.mean(gradient_b)
    return gradient_w, gradient_b
```

gradient()函数计算梯度。训练数据的关键是找到一组(w,b)，使得损失函数loss取最小值。一般采用梯度下降法，即沿着梯度的反方向是函数值下降最快的方向。根据梯度计算公式，使用numpy矩阵操作对所有权重参数w_j (j=0,1,...,12)操作，一次性计算出13个参数对应的梯度值。对于N个样本的情况，可以直接使用代码第17行计算出所有样本对梯度的贡献。显然，这里利用扩展库numpy的广播功能使得矩阵计算更加便捷。

$$\frac{\partial \mathrm{L}}{\partial w_j}=\frac{1}{N}\sum_i^N(z^{(i)}-y^{(i)})\frac{\partial z_j^{(i)}}{w_j}=\frac{1}{N}\sum_i^N(z^{(i)}-y^{(i)})\,x_j^{(i)}$$

变量gradient_w的每一行代表一个样本对梯度的贡献。根据梯度计算公式，总梯度是样本对梯度贡献的平均值。代码第18行使用np.mean()函数得到梯度贡献的均值，相当于将矩阵的每一行相加之后除以总行数。

波士顿房价预测

The prediction of house price in Boston

利用Python构建网络模型进行房价预测

Step 1　网络模型构建

```
def gradient(self, x, y):
    z = self.forward(x)
    gradient_w = (z-y)*x
    gradient_w = np.mean(gradient_w, axis=0)
    gradient_w = gradient_w[:, np.newaxis]
    gradient_b = (z-y)
    gradient_b = np.mean(gradient_b)
    return gradient_w, gradient_b
```

代码第19行中[:, np.newaxis]意为对全部数据执行增加一个维度的操作，以适应代码中矩阵操作的要求。由于np.mean()函数消除了第0维，导致使用numpy矩阵操作完成梯度计算的同时引入了一个新问题：gradient_w的形状是(13,)，而w的维度为(13, 1)。为了计算方便，gradient_w和w必须保持一致的形状。因此代码第19行使用np.newaxis()函数增加一个维度，将一维的gradient_w转换成一个维度为(13, 1)的二维矩阵，便于后面的矩阵操作。

波士顿房价预测

The prediction of house price in Boston

利用Python构建网络模型进行房价预测

Step 1　网络模型构建

```
def update(self, gradient_w, gradient_b, eta = 0.01):
        self.w = self.w - eta * gradient_w
        self.b = self.b - eta * gradient_b
```

update()函数实现梯度更新。沿着梯度的反方向移动eta * gradient_w的距离，更新参数w和b。

波士顿房价预测

The prediction of house price in Boston

利用Python构建网络模型进行房价预测

Step 1　网络模型构建

```
def train(self, x, y, iterations=100, eta=0.01):
    losses = []
    for i in range(iterations):
        z = self.forward(x)
        L = self.loss(z, y)
        gradient_w, gradient_b = self.gradient(x, y)
        self.update(gradient_w, gradient_b, eta)
        losses.append(L)
        if (i+1) % 10 == 0:
            print('iter {}, loss {}'.format(i, L))
    return losses
```

train()函数完成模型训练。按照定义的移动步长eta，前向传播计算预测值z，计算损失值loss，通过print()函数输出损失函数的变化，每次沿着梯度的反方向移动一小步，到达下一个点，观察损失函数的变化并返回损失值。

波士顿房价预测

The prediction of house price in Boston

利用Python构建网络模型进行房价预测

完成模型网络结构Network类的定义之后，下面利用网络模型进行房价预测。首先获取数据，然后训练网络，最后进行房价预测。

Step 2　利用网络模型进行房价预测

```
# 获取数据
train_data, test_data = load_data()
x = train_data[:, :-1]
y = train_data[:, -1:]
```

用于加载数据：取所有样本中前13列为特征值，保存于变量x；最后一列为目标值，保存于变量y。

```
# 创建网络
net = Network(13)
num_iterations=1000
# 启动训练
losses = net.train(x,y, iterations=num_iterations, eta=0.01)
```

用于构建网络并启动训练：网络权重变量设置为13个，迭代次数设置为1000，学习率设置为0.01。

波士顿房价预测

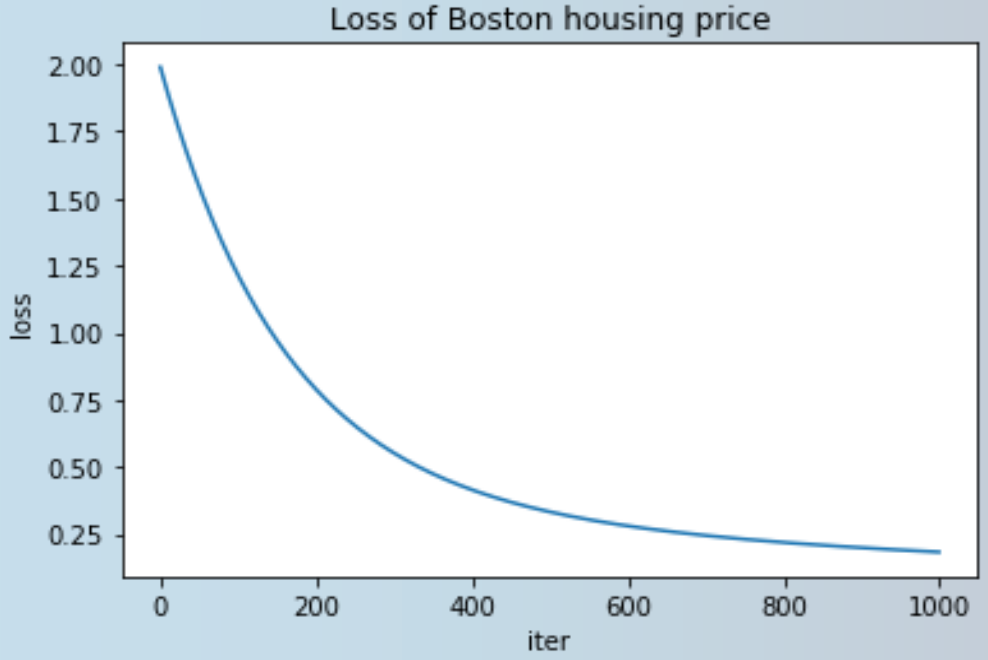
The prediction of house price in Boston

利用Python构建网络模型进行房价预测

Step 2　利用网络模型进行房价预测

```
# 画出损失函数的变化趋势
plot_x = np.arange(num_iterations)
plot_y = np.array(losses)
plt.plot(plot_x, plot_y)
plt.title("Loss of Boston housing price")
plt.xlabel("iter")
plt.ylabel("loss")
plt.show()
```

绘制损失函数变化趋势图：每迭代10次绘制一次损失值，生成散点图。



波士顿房价预测

The prediction of house price in Boston

模型评估

与分类预测结果相比，回归预测的数值型结果不能严苛地与真实值完全相同。一般情况下，人们可以使用多种评价函数衡量预测值与真实值之间的差距，如平均绝对误差（MAE）和均方误差（MSE），但是这两种评价指标与具体的应用场景有关。在不同应用场景中评价模型的可比性时，这两种评价函数存在缺陷；R-squared评价既考虑回归值与真实值之间的差异，又兼顾应用场景中真实值本身的变动，在统计含义上表现出模型在数值回归方面的拟合能力。

波士顿房价预测

The prediction of house price in Boston

01

线性回归器性能评估

模型评估

首先，分别使用三种回归评价机制和两种调用R-squared评价模块的方法对线性回归器的性能进行评估。

使用linearregression 模块自带的评估模块

```
print('The value of default means of LinearRegression is ', \
 lr.score(x_test, y_test))
```

波士顿房价预测

The prediction of house price in Boston

模型评估

01

线性回归器性能评估

使用专门的评价模块进行评估

```
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.metrics import mean_absolute_error
print('The value of R-squared of LinearRegression is ',\
      r2_score(y_test, lr_y_pred))
print('The mean squared error of LinearRegression is ',
      mean_squared_error(ss_y.inverse_transform(y_test), \
       ss_y.inverse_transform(lr_y_pred)))
print('The mean absolute error of LinearRegression is ',
      mean_absolute_error(ss_y.inverse_transform(y_test), \
       ss_y.inverse_transform(lr_y_pred)))
```

波士顿房价预测

The prediction of house price in Boston

模型评估

01

线性回归器性能评估

使用SGDRegressor 自带模块进行评估

```
print('The value of default measurement of SGDRegressor is ',\
 sgdr.score(x_test, y_test))
```

波士顿房价预测

The prediction of house price in Boston

模型评估

01

线性回归器性能评估

使用专门的评价模块进行评估

```
print('The value of R-squared of SGDRegressor is ', \
      r2_score(y_test, sgdr_y_pred))
print('The mean squared error of SGDRegressor is ',
      mean_squared_error(ss_y.inverse_transform(y_test), \
       ss_y.inverse_transform(sgdr_y_pred)))
print('The mean absolute error of SGDRegressor is ',
      mean_absolute_error(ss_y.inverse_transform(y_test),\
ss_y.inverse_transform(sgdr_y_pred)))
```

波士顿房价预测

The prediction of house price in Boston

模型评估

01

线性回归器性能评估

代码运行结果如下：

The value of default means of LinearRegression is 0.675795501452948 1

The value of R-squared of LinearRegression is 0.675795501452948 1

The mean squared error of LinearRegression is 25.139236520353453

The mean absolute error of LinearRegression is 3.532532543705398

The value of default measurement of SGDRegressor is 0.6692921751176728

The value of R-squared of SGDRegressor is 0.6692921751176728

The mean squared error of SGDRegressor is 25.643512863353674

The mean absolute error of SGDRegressor is 3.49695922108399

波士顿房价预测

The prediction of house price in Boston

01

线性回归器性能评估

模型评估

```
① # 1.使用linearregression模块自带的评估模块
② print('The value of default means of LinearRegression is ', \
③ lr.score(x_test, y_test))
④ # 2.使用专门的评价模块进行评估
⑤ from sklearn.metrics import r2_score, mean_squared_error
⑥ from sklearn.metrics import mean_absolute_error
⑦ print('The value of R-squared of LinearRegression is ',\
⑧         r2_score(y_test, lr_y_pred))
```

输出结果为：The value of default means of LinearRegression is 0.675795501452948 1

从代码2~3行和代码7~8行的输出结果可以看出，回归模型自带的评估模块与sklearn.metrics中r2_score()评估函数得到的结果相同，可见这两种调用R-squared的方式是等价的。

波士顿房价预测

The prediction of house price in Boston

01

线性回归器性能评估

模型评估

```
① # 1.使用linearregression模块自带的评估模块
② print('The value of default means of LinearRegression is ', \
③ lr.score(x_test, y_test))
④ # 2.使用专门的评价模块进行评估
⑤ from sklearn.metrics import r2_score, mean_squared_error
⑥ from sklearn.metrics import mean_absolute_error
⑦ print('The value of R-squared of LinearRegression is ',\
⑧       r2_score(y_test, lr_y_pred))
⑨ print('The mean squared error of LinearRegression is ',
⑩       mean_squared_error(ss_y.inverse_transform(y_test), \
⑪ ss_y.inverse_transform(lr_y_pred)))
⑫ print('The mean absolute error of LinearRegression is ',
⑬       mean_absolute_error(ss_y.inverse_transform(y_test), \
⑭ ss_y.inverse_transform(lr_y_pred)))
```

输出结果：

The value of default means of LinearRegression is 0.675795501452948 1

The value of R-squared of LinearRegression is 0.675795501452948 1

The mean squared error of LinearRegression is 25.139236520353453

The mean absolute error of LinearRegression is 3.532532543705398

代码1~14行的输出结果表明，三种评价方式R-squared、MSE和MAE在评估结果的具体取值上不同，但是在评价总体优劣程度的趋势上基本一致。

波士顿房价预测

The prediction of house price in Boston

01

线性回归器性能评估

模型评估

```
# 3.使用SGDRegressor自带模块进行评估
print('The value of default measurement of SGDRegressor is ',\
 sgdr.score(x_test, y_test))
# 4.使用专门的评价模块进行评估
print('The value of R-squared of SGDRegressor is ', \
 r2_score(y_test, sgdr_y_pred))
print('The mean squared error of SGDRegressor is ',
       mean_squared_error(ss_y.inverse_transform(y_test), \
         ss_y.inverse_transform(sgdr_y_pred)))
print('The mean absolute error of SGDRegressor is ',
       mean_absolute_error(ss_y.inverse_transform(y_test),\
ss_y.inverse_transform(sgdr_y_pred)))
```

输出结果：

The value of default measurement of SGDRegressor is 0.669292175117672 8

The value of R-squared of SGDRegressor is 0.669292175117672 8

The mean squared error of SGDRegressor is 25.643512863353674

The mean absolute error of SGDRegressor is 3.4969592210839 9

与代码1-14行输出结果对比，使用随机梯度下降的方法SGDRegressor进行参数估计在性能上不如使用解析的方法LinearRegression。然而，在数据规模庞大的模型训练任务中，随机梯度下降法具有节省计算时间的优势。

02

支持向量机能性评估

模型评估

类似地，使用R-squared、MSE和MAE指标对三种配置的支持向量机回归模型在相同测试集上进行性能评估。代码如下：

```
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.metrics import mean_squared_error
print('R-squared of linear SVR:',linear_svr.score(x_test,y_test))
print('The mean squared error of linear SVR is ', \
      mean_squared_error(ss_y.inverse_transform(y_test), \
      ss_y.inverse_transform(linear_svr_y_pred)))
print('The mean absolute error of linear SVR is ', \
      mean_absolute_error(ss_y.inverse_transform(y_test),\
      ss_y.inverse_transform(linear_svr_y_pred)))
print('\n')
print('R-squared of poly SVR is', poly_svr.score(x_test, y_test))
print('The mean squared error of poly SVR is ', \
      mean_squared_error(ss_y.inverse_transform(y_test), \
      ss_y.inverse_transform(poly_svr_y_pred)))
print('The mean absolute error of poly SVR is ', \
      mean_absolute_error(ss_y.inverse_transform(y_test),\
      ss_y.inverse_transform(poly_svr_y_pred)))
print('\n')
print('R-squared of RBF SVR is', rbf_svr.score(x_test,y_test))
print('The mean squared error of RBF SVR is ', \
      mean_squared_error(ss_y.inverse_transform(y_test), \
      ss_y.inverse_transform(rbf_svr_y_pred)))
print('The mean absolute error of RBF SVR is ', \
      mean_absolute_error(ss_y.inverse_transform(y_test),\
      ss_y.inverse_transform(rbf_svr_y_pred)))
```

波士顿房价预测

The prediction of house price in Boston

02

支持向量机能性评估

模型评估

代码运行结果如下：

```
R-squared value of linear SVR is  0.650659546421538
The mean squared error of linear SVR is  27.088311013556027
The mean absolute error of linear SVR is  3.4328013877599624

R-squared value of poly SVR is  0.403650651025512
The mean squared error of poly SVR is  46.241700531039
The mean absolute error of poly SVR is  3.7384073710465047

R-squared value of RBF SVR is  0.7559887416340946
The mean squared error of RBF SVR is  18.920948861538715
The mean absolute error of RBF SVR is  2.6067819999501114
```

波士顿房价预测

The prediction of house price in Boston

模型评估

02

支持向量机能性评估

三组性能评测的结果表明，不同配置的模型在相同测试集上的性能表现存在显著差异。使用径向基核函数对特征进行非线性映射后，支持向量机表现出最佳的回归性能。

同理，对不同配置的K近邻回归模型在房价预测任务中的性能评估结果表明，采用加权平均的K近邻回归策略具有更好的预测性能；默认配置的回归树在测试集上的性能优于上述两种线性回归器；三种集成回归模型在“波士顿房价”数据集上获得了更好的预测性能和更强的稳定性，然而它们在模型训练中耗费的时间也更多。