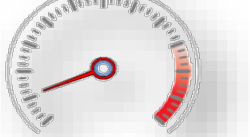
Python与大数据分析

--大数据分析-预处理

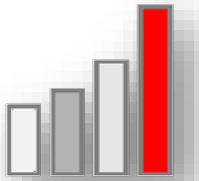
大数据的概念（技术特点）

- 数据量大
- 数据类型繁多
- 处理速度快
- 价值密度低



VELOCITY

快速化



VOLUME

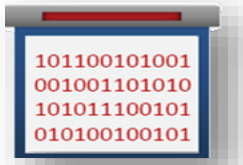
大量化

4V

VARIETY

多样化

价值密度低



VALUE

Veracity：数据的准确性和可信赖度，即数据的质量。

从数据量角度，大数据泛指无法在可容忍的时间内用传统信息技术和软硬件工具对其进行获取、管理和处理的巨量数据集，需要可伸缩的计算体系结构

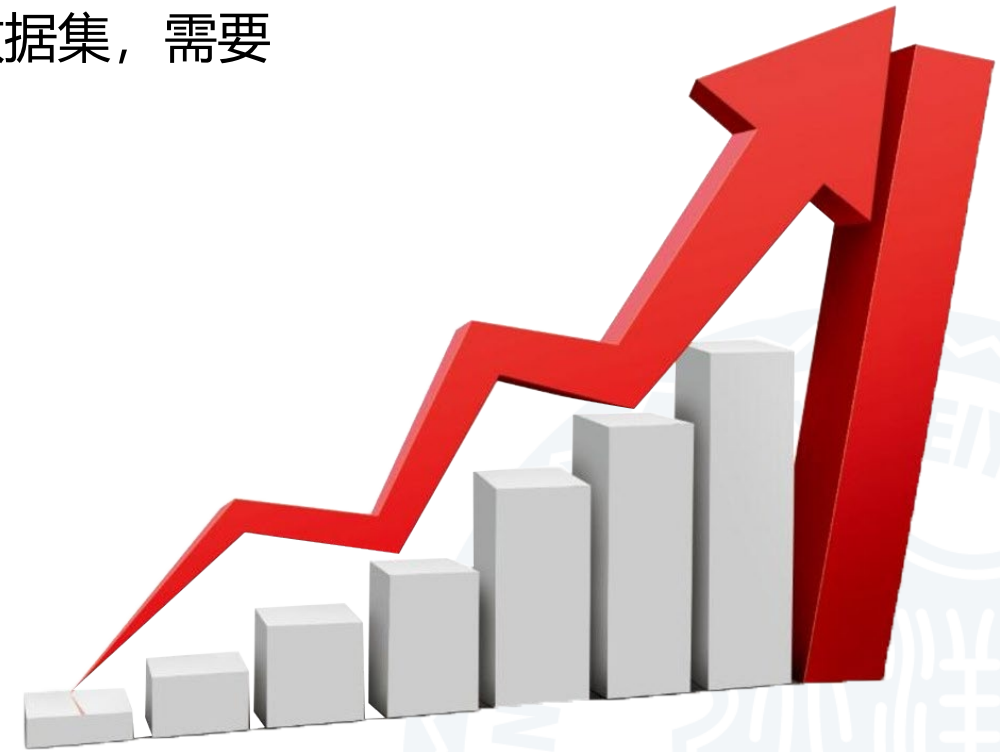
根据IDC发布报告

人类社会数据每年

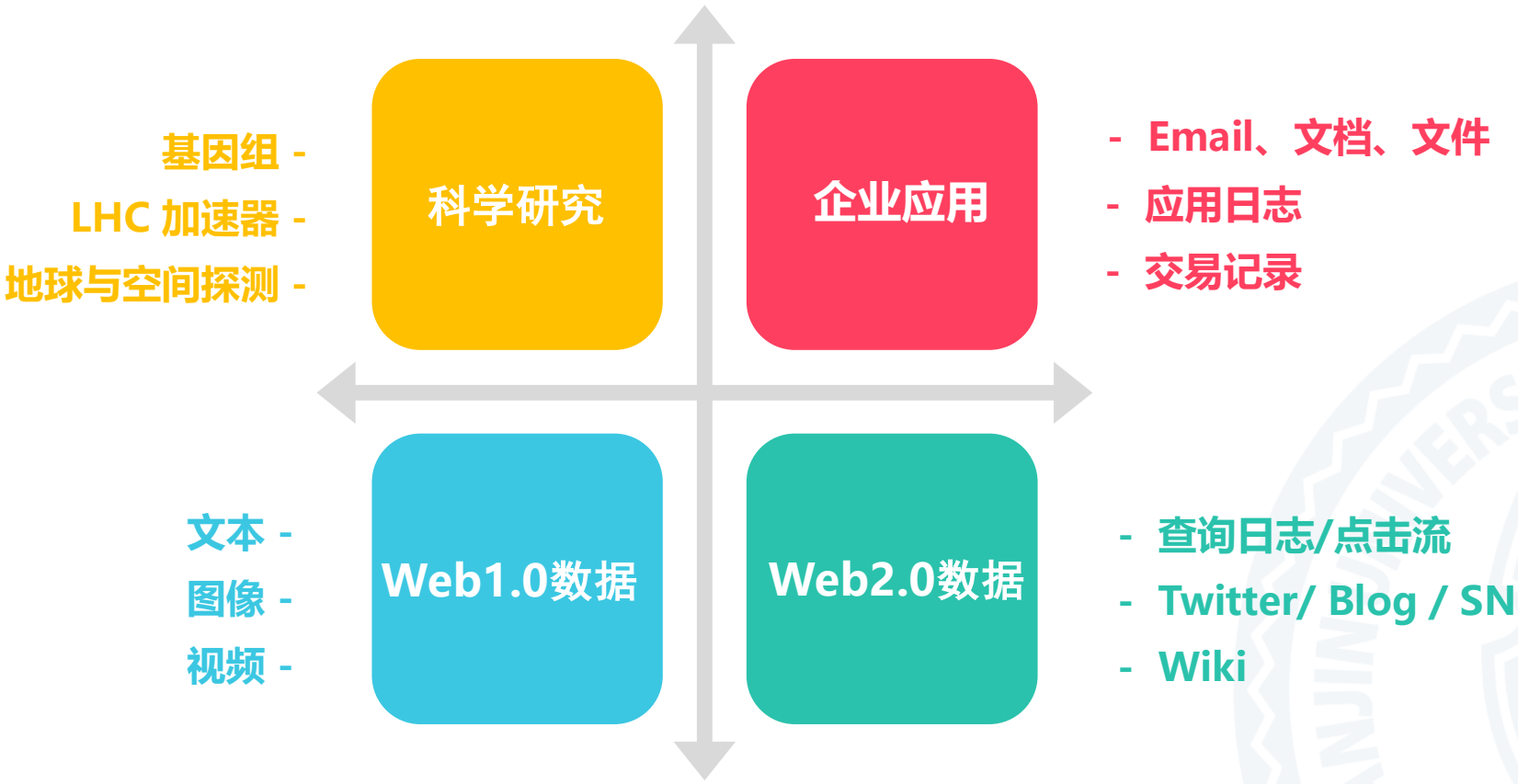
50%的速度，**每两**

年就增长一倍

大数据摩尔定律

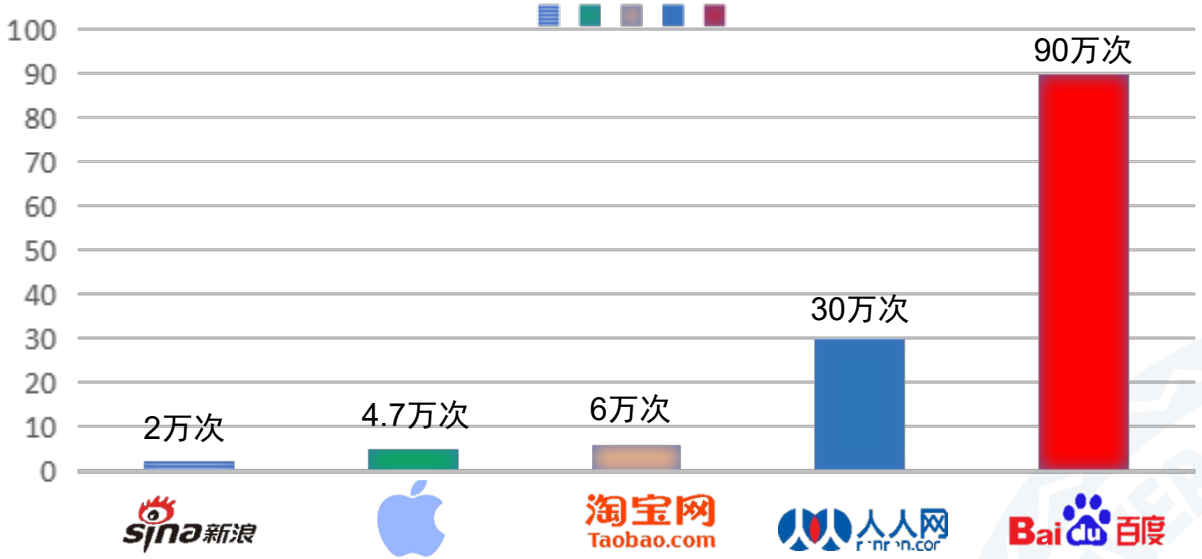


数据类型繁多


基因组 -
LHC 加速器 -
地球与空间探测 -
科学研究
企业应用
- Email、文档、文件
- 应用日志
- 交易记录
文本 -
图像 -
视频 -
Web1.0数据
Web2.0数据
- 查询日志/点击流
- Twitter/ Blog / SNS
- Wiki

处理速度快　快速化　VELOCITY

1秒定律



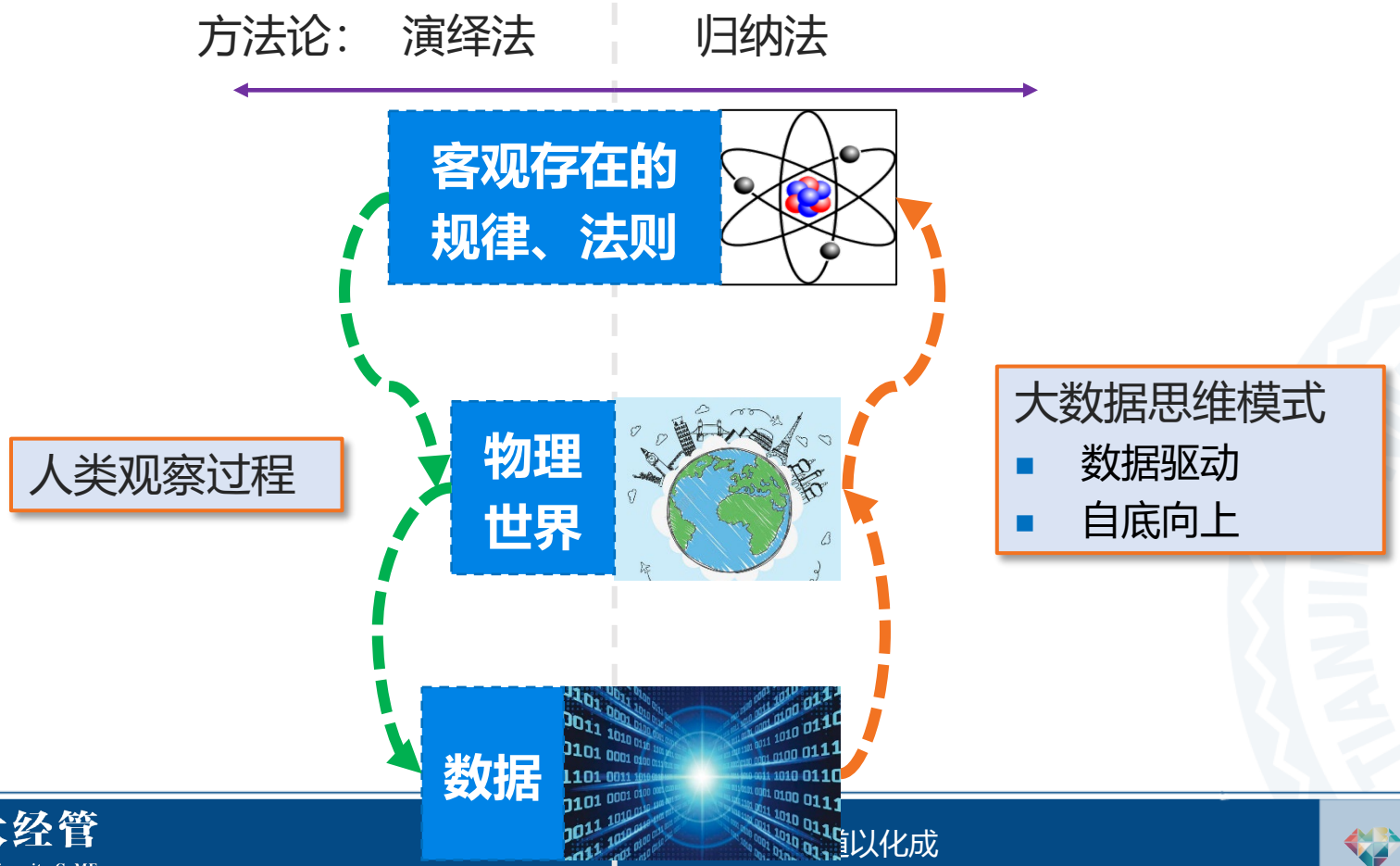
一秒定律　从数据生成到决策响应仅需1秒

价值密度低

价值密度低

大数据是系统性方法论

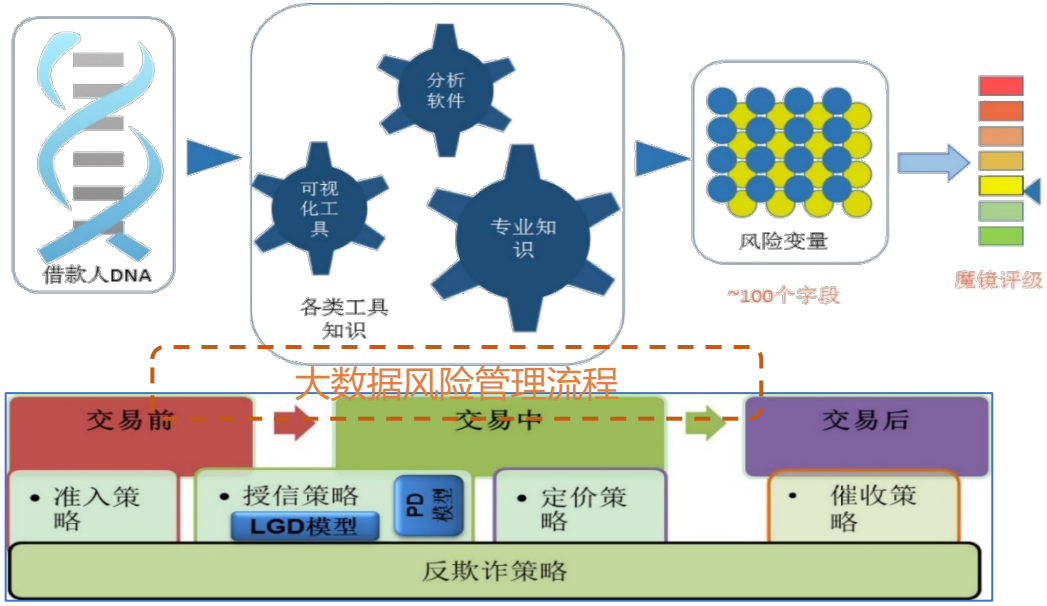
- 大数据从数据驱动、自底向上的角度解决问题
- 一切皆可数据化



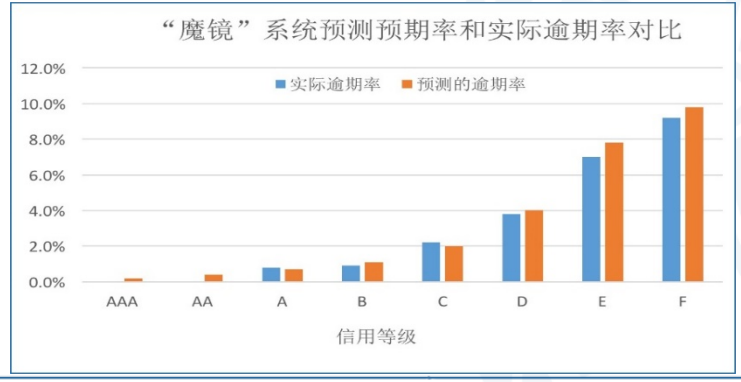
大数据的力量

案例1: 大数据风控1

- 魔镜系统是拍拍贷自主开发、具有自主知识产权的风险评估系统，其核心是一系列基于大数据的风控模型。针对每一笔借款，风险模型会给出一个风险评分，以反映对逾期率的预测。每一个评分区间会以一个字母评级的形式展示给借入者和借出者。从AA到F，风险依次上升。



魔镜评级	目标逾期率
A	<0.5%
B	0.5-1%
C	1-2%
D	2-4%
E	4-8%
F	>10%



大数据的力量

案例2: 大数据风控2

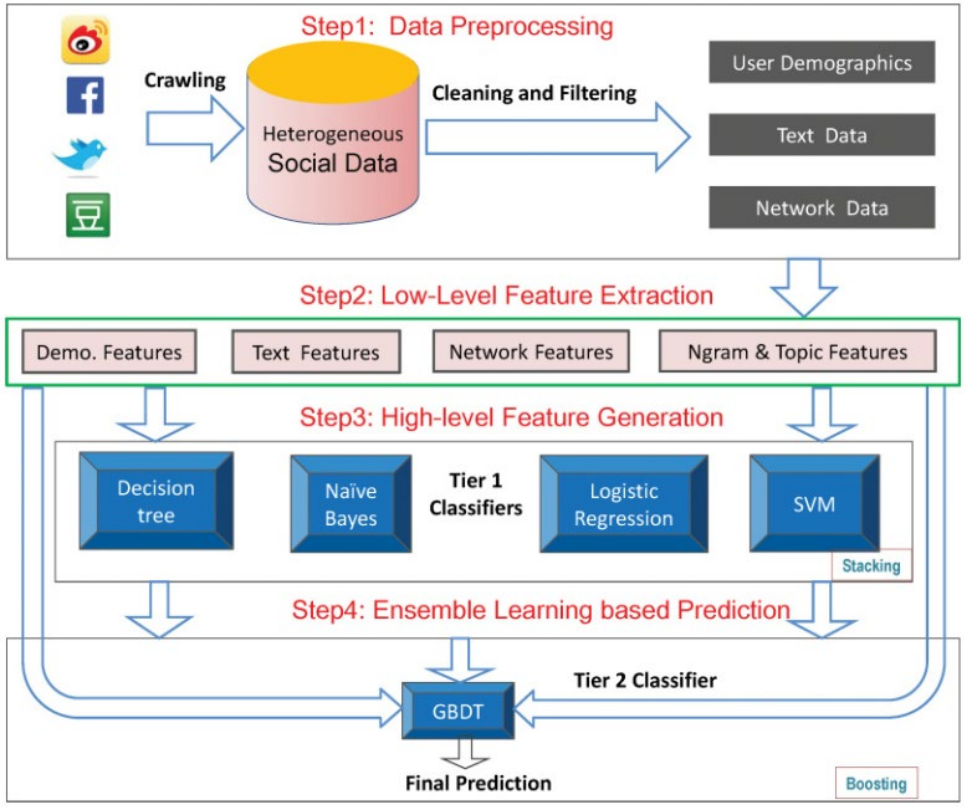


Fig. 4. Illustration of our two-tier ensemble learning framework.

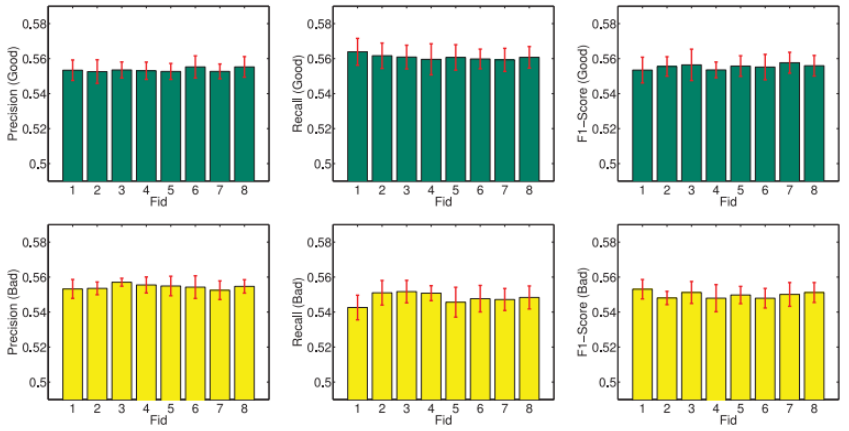


Fig. 7. Prediction results for both good and bad credit users with different features in Table VII as the input.

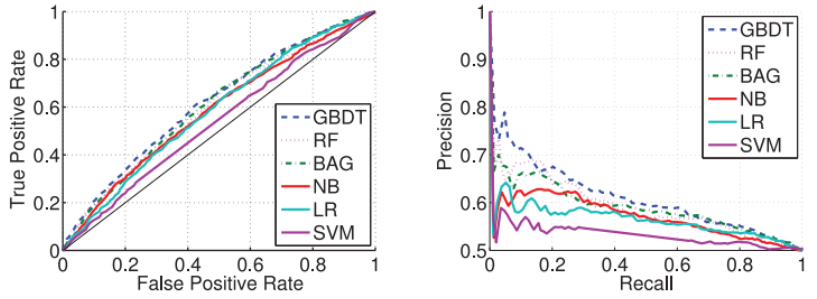
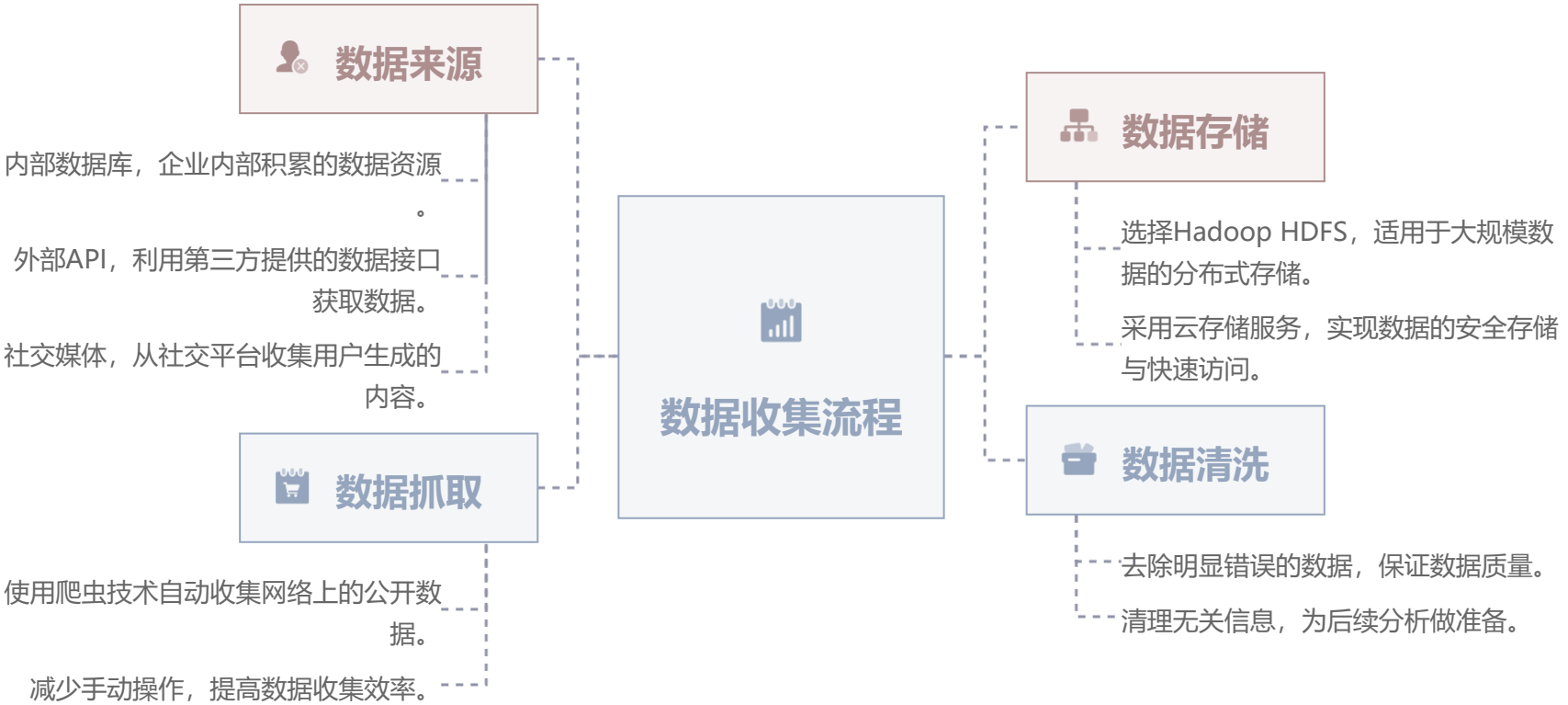


Fig. 10. ROC and Precision-Recall curves of different learning algorithms with LF+HF as input.

大数据分析的流程

- 数据收集
- 数据处理
- 数据分析

数据收集的流程


数据来源
内部数据库，企业内部积累的数据资源。
外部API，利用第三方提供的数据接口获取数据。
社交媒体，从社交平台收集用户生成的内容。
数据抓取
使用爬虫技术自动收集网络上的公开数据。
减少手动操作，提高数据收集效率。
数据收集流程
数据存储
选择Hadoop HDFS，适用于大规模数据的分布式存储。
采用云存储服务，实现数据的安全存储与快速访问。
数据清洗
去除明显错误的数据，保证数据质量。
清理无关信息，为后续分析做准备。

数据收集-公开数据集

公开数据集

利用政府、科研机构及企业公开的数据集，如Kaggle、UCI Machine Learning Repository等，是获取高质量数据的常见途径。

国家统计局 National Bureau of Statistics

[中国统计年鉴 - 国家统计局]

kaggle

[Kaggle: Your Machine Learning and Data Science Community]

数据收集

API接口

许多平台提供API供开发者调用，直接获取实时数据，如Twitter API、Google Maps API等，高效且准确。

幂简集成 - 品种超全的API接口平台, 一站搜索、试用、集成国内外API接口

Nokia acquires Rapid technology and team!

接口大全-免费API,收集所有免费的API

数据收集-网络爬虫

网络爬虫

通过编写爬虫程序从网站抓取数据，适用于社交媒体、新闻、论坛等非结构化信息的收集。

网络爬虫概念

网络爬虫是自动抓取网页信息的程序，用于大规模数据收集。

爬虫工作原理

通过URL访问网页，解析HTML，提取所需数据，存储并跟踪链接进行深度搜索。

爬虫类型

通用爬虫覆盖广泛网站，聚焦爬虫针对特定主题或领域。

爬虫法律与伦理

遵守robots.txt规则，尊重版权，避免高频请求影响网站正常运行。

数据收集-网络爬虫

我们必须要理解网页的基本构造与组成，然后才能去分析筛选出需要的数据。

无论是什么样的网页，都必须是这样的布局。**HTML是支撑网页内容的部分**。通常来讲，我们在浏览器中看到的网页内容就是浏览器解析超文本后的输出结果，即HTML。**服务器将HTML文档返回给客户端之后，我们使用的浏览器是知道HTML语法的，所以它会自动解析。**解析完之后就是我们所看到的页面。能够实现比文本更丰富的内容。

数据收集-网络爬虫

```html
<!DOCTYPE html>
<html>
    <head>
        <!-- head 标签中的内容不会在浏览器窗口中显示 -->
        <title>这是页面标题</title>
    </head>
    <body>
        <!-- body 标签中的内容会在浏览器窗口中显示 -->
        <h2>这是一级标题</h2>
        <p>这是一段文本</p>
    </body>
</html>
```

标签、层叠样式表（CSS）和 JavaScript 是构成 HTML 页面的三个基本组成部分。**标签用来承载页面要显示的内容，CSS 负责对页面的渲染，而 JavaScript 用来控制页面的交互式行为。**要实现 HTML 页面的解析，可以使用 **XPath 的语法**，它原本是 XML 的一种查询语法，可以根据 HTML 标签的层次结构提取标签中的内容或标签属性；此外，也可以使用 CSS 选择器来定位页面元素，就跟用 CSS 渲染页面元素是同样的道理。

数据收集-网络爬虫

html是以标签为单位的，不同的表标签提供不同的内容

- 双标签:
  - <标签名 属性1=属性值1 属性2=属性值2 .....> 标签和内容 </标签名>
- 单标签：
  - <标签名 属性名1=属性值1 属性名2=属性值2…>
  - <标签名 属性名1=属性值1 属性名2=属性值2…/>

数据收集-网络爬虫

- 使用h1到h6表示不同级别的标题，其中h1级别的标题字体最大，h6级别的标题字体最小：

```
<h1>一级标题</h1>
<h2>二级标题</h2>
<h3>三级标题</h3>
```

- p标签表示段落：

```
<p>这是一个段落</p>
```

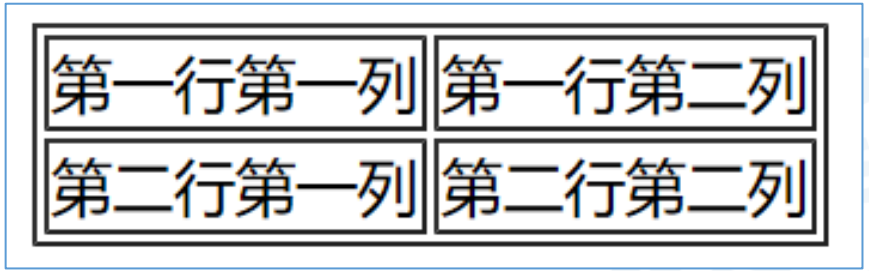
- a标签表示超链接，使用时需要指定链接地址（由href属性来指定）和在页面上显示的文本：

```
<a href="http://www.baidu.com">点这里</a>
```

数据收集-网络爬虫

- table、tr、td标签 ——table标签用来创建表格，tr用来创建行，td用来创建单元格：

```
<table border="1">
  <tr>
    <td>第一行第一列</td>
    <td>第一行第二列</td>
  </tr>
  <tr>
    <td>第二行第一列</td>
    <td>第二行第二列</td>
  </tr>
</table>
```



数据收集-网络爬虫

- ul、ol、li ——ul标签用来创建无序列表，ol标签用来创建有序列表，li标签用来创建其中的列表项：

```
<ul id="colors" name="myColor">
  <li>红色</li>
  <li>绿色</li>
  <li>蓝色</li>
</ul>
```

数据收集-网络爬虫

- div标签用来创建一个块，其中可以包含其他标签：

```
<div id="yellowDiv" style="background-
    color:yellow;border:#FF0000 1px solid;">
    <ol>
        <li>红色</li>
        <li>绿色</li>
        <li>蓝色</li>
    </ol>
</div>
<div id="reddiv" style="background-color:red">
    <p>第一段</p>
    <p>第二段</p>
</div>
```

网络爬虫

CSS 语法

- 选择器{属性值1：属性值；属性2：属性值2；…}
- 选择器：选中需要添加样式的标签。

CSS 选择器

- 元素选择（标签选择） ----- 将标签作为选择器，选择所有指定标签
  - p{ } 选中所有的p标签
  - a{ } - 选择所有的a标签
- id标签 ----- 在id属性值前加#作为一个选择器，选择id属性值为指定值的标签 每一个可见的标签都可以设置id属性，并且同一个页面中，同一个id值只有一标签
  - #p1 - 选择id 属性值为p1的标签
  - #a - 选择id 属性值为a的标签

网络爬虫

CSS 选择器

- class 选择器 ---- 在class属性值前面加.作为一个选择器，选择class属性值为指定值的标签
  - 不同的标签可以有相同的class值；同一个标签可以有不同的class值
  - class值有多个，多个值之间用一个空格隔开，class选择器获取标签类型
  - c1 - 获取class值为c1的所有标签
  - p.c1 - 获取class值为c1的所有p标签
  - .c1.c2 - 获取class值同时为c1和c2的标签

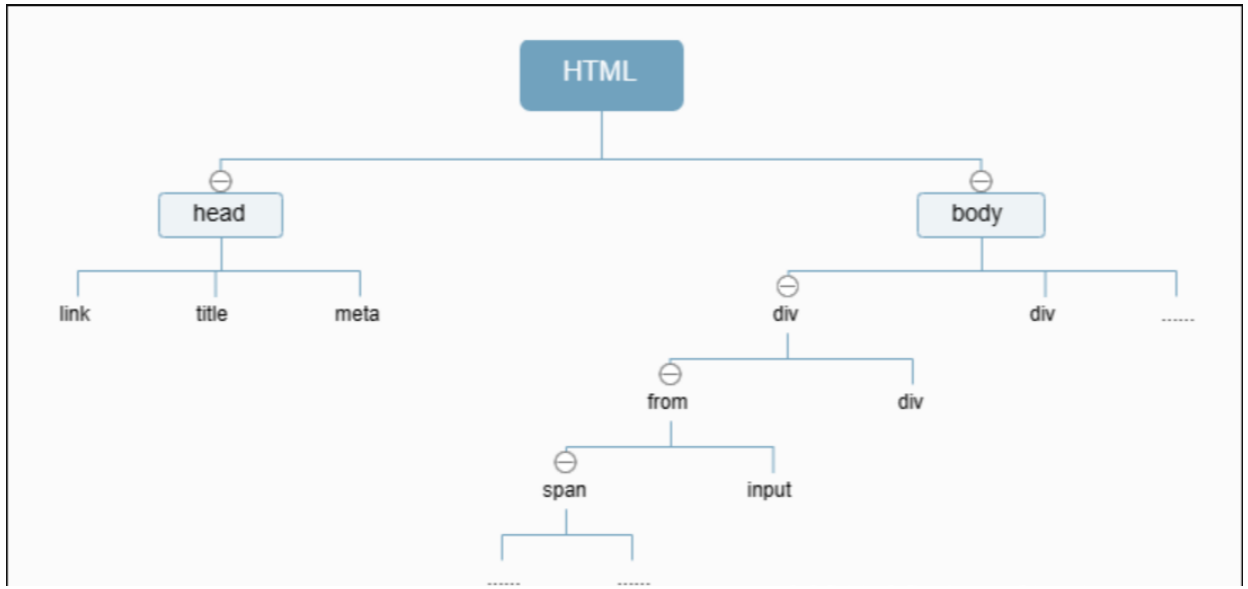
网络爬虫

CSS 选择器

- 群组选择器 - 将多个选择器用逗号隔开作为一个选择器
  - p , a{} - 选择所有的p标签和所有的a标签
  - #p1, c1 , p{} - 选中id为 #p1 的标签和class为c1的标签，和所有的p标签
- 子代选择器 - 多个选择器用>隔开作为一个选择器
  - div>{ }
- 后代选择器 - 多个选择器用空格隔开作为一个选择器
  - div p{ } (之间有一个空格)

网络爬虫-XPath

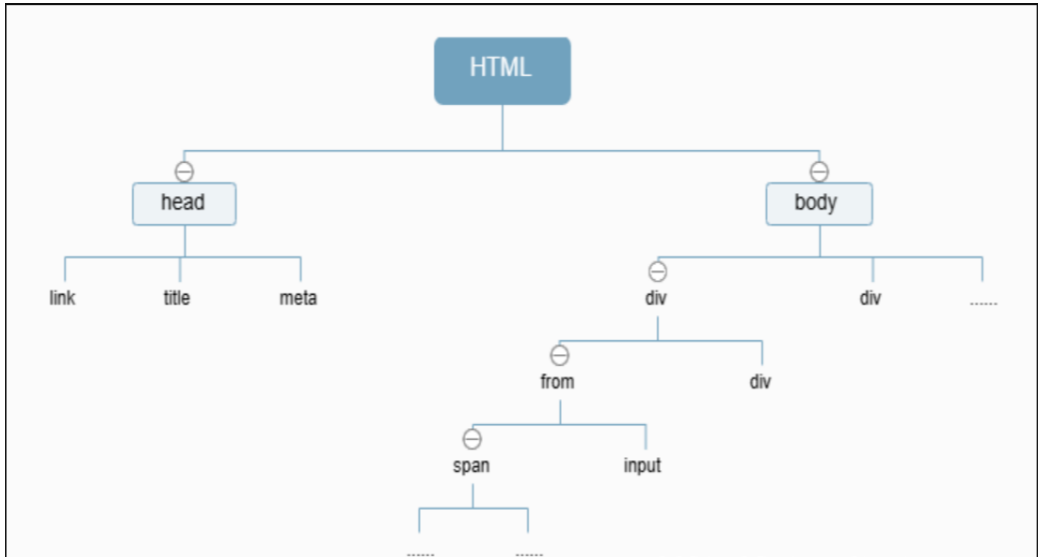
- XPath（XML Path Language）是一种XML（可拓展标记语言）的查询语言，他能在XML树状结构中寻找节点。XPath 用于在 XML 文档中通过元素和属性进行导航。可以方便的定位xml中的元素和其中的属性值。



注意： HTML 的结构就是树形结构，HTML 是根节点，所有的其他元素节点都是从根节点发出的。其他的元素都是这棵树上的节点Node，每个节点还可能有属性和文本。而路径就是指某个节点到另一个节点的路线。

而我们使用xpath就是根据这些路线所确定的。

网络爬虫-节点之间的关系



- 父节点(Parent)： HTML 是 body 和 head 节点的父节点；
- 祖先节点(Ancestor)：body 是 form 的祖先节点，爷爷辈及以上；
- 子节点(Child)：head 和 body 是 HTML 的子节点；
- 后代节点(Descendant)：form 是 HTML 的后代节点，孙子辈及以下。
- 兄弟节点(Sibling)：拥有相同的父节点，head 和 body 就是兄弟节点。title 和 div 不是兄弟，因为他们不是同一个父节点。

网络爬虫-lxml库

- lxml是python中的一个第三方模块，它包含了将html文本转成xml对象，和对对象执行xpath的功能。

```
>>> pip install lxml

>>> from lxml import etree

>>> tree= etree.HTML(网页源代码)

>>> tree.xpath(一段神奇的符号) # 返回的数据类型是列表，可以通过索引指定获取，或循环取出每个元素!
```

- XPath语法
  - 层级：/ 直接子级、// 跳级
  - 属性：@属性访问
  - 函数：contains()、text() 等

网络爬虫-lxml库

语法	说明	实例
//	99%的情况下	//div
/	用/来选择子元素	//div/span
//**//	选择//的后代元素	//div//span
[]与@	用于标签后添加筛选条件，@符号通过元素的属性实现实施	//div[@class= ‘example’]
*	适配所有的元素	//div/*
Text()	选择拥有特定的文本名称	//div/p[text()= ‘tianjing’]

网络爬虫-lxml库

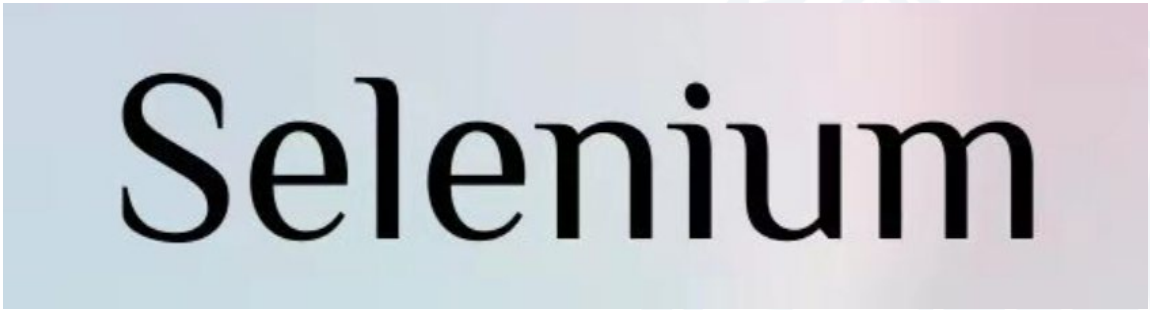
语法	实例	实例
and	//input[@id='kw' and @class='s_ipt']	input标签下i的属性为kw和class为s_ipt
or	//input[@id='kw' or @class='s_t']	input标签下i的属性为kw或者class为s_t
!=	//input[@class!='1111']	input标签下i的属性class不为1111
not	//year[not(.=2005)]	year标签下的text不为2055
<,>	//div[@class="cell" and text()>'1336'] //div[@class="cell" and text()<'1336']	div标签下class值为cell，text内容大于1336
|	//book/title | //book/price	book标签下title和price标签

网络爬虫-Selenium库

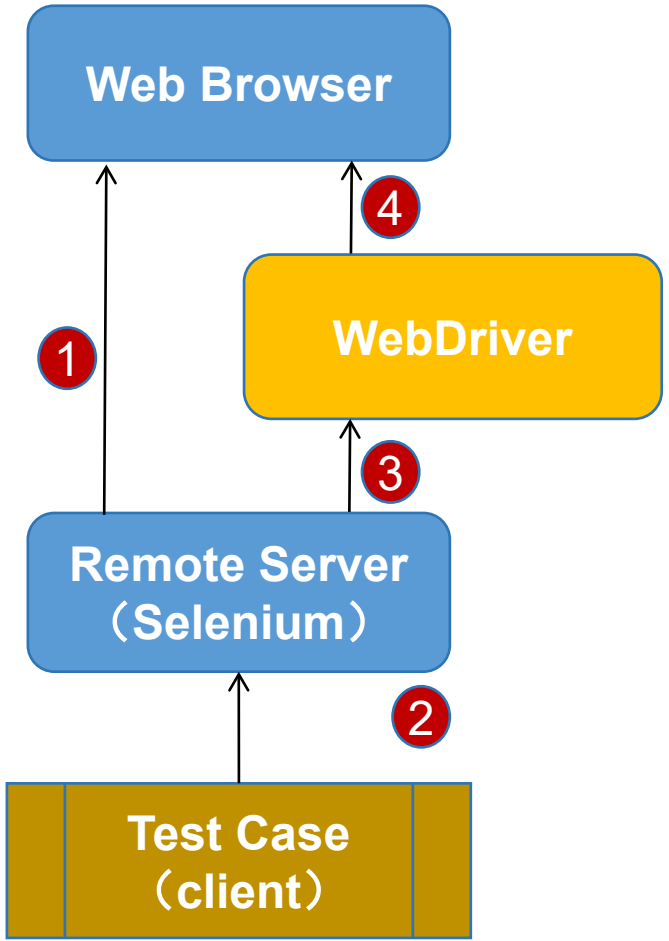
- Selenium 测试脚本可以使用任何支持的编程语言进行编码，并且可以直接在大多数现代 Web 浏览器中运行。在爬虫领域 selenium 同样是一把利器，能够解决大部分的网页的反爬问题。

- 是最广泛使用的开源 Web UI（用户界面）自动化测试套件。

```
>>> pip install selenium
>>> import requests
```



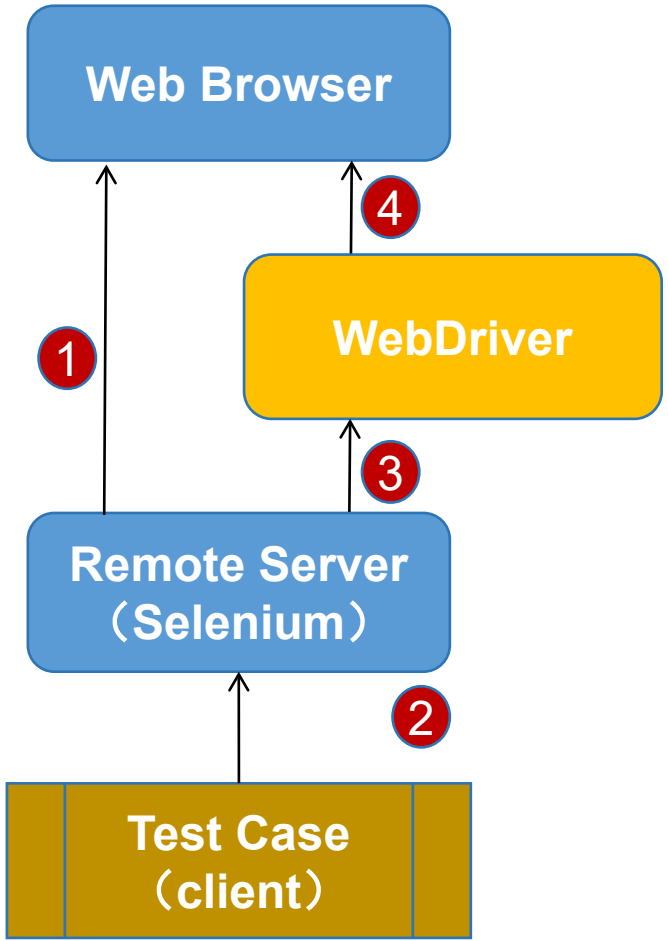
网络爬虫-Selenium库



- ✓ 当使用 Selenium启动浏览器时，后台会同时启动基于 WebDriver Wire 协议的 Web Service 作为 Selenium 的 Remote Server，并与浏览器绑定。之后，Remote Server 就开始监听 Client 端的操作请求；

- ✓ 执行测试时，测试用例会作为 Client 端，将需要执行的页面操作请求以 Http Request 的方式发送给 Remote Server 。 该 Http Request 的 body，是以 WebDriver Wire 协议规定的 JSON 格式来描述需要浏览器执行的具体操作；

网络爬虫-Selenium库

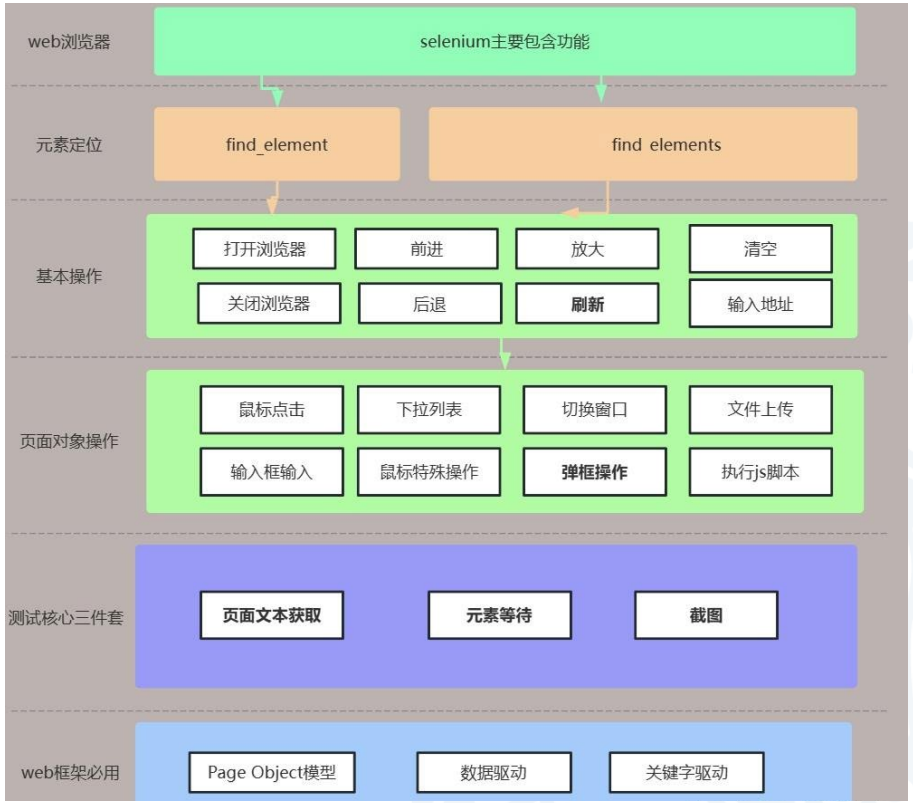


- ✓ Remote Server 接收到请求后，会对请求进行解析，并将解析结果发给WebDriver，由WebDriver 实际执行浏览器的操作；
- ✓ WebDriver 可以看做是直接操作浏览器的原生组件（Native Component），所以搭建测试环境时，通常都需要先下载浏览器对应的 WebDriver。

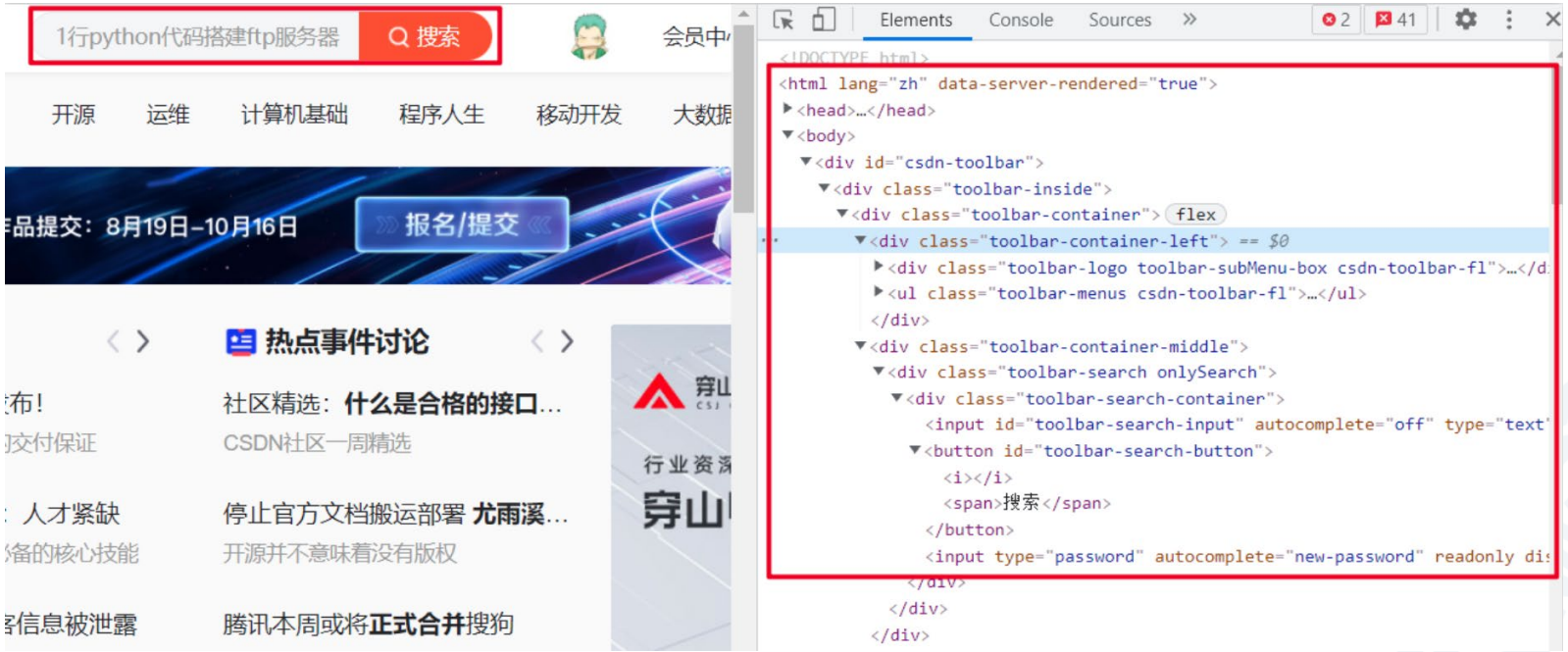
网络爬虫-Selenium库

针对不同的浏览器，需要安装不同的驱动

- **Firefox** 浏览器驱动：Firefox
- **Chrome** 浏览器驱动：ChromeDriver
- **IE** 浏览器驱动：IE
- **Edge** 浏览器驱动：Edge
- **PhantomJS** 浏览器驱动：PhantomJS
- **Opera** 浏览器驱动：Opera



网络爬虫-Selenium库



想要定位并获取页面中的信息，首先要使用 webdriver 打开指定页面，再去定位。

```
>>> browser.get('https://www.csdn.net/')
```

网络爬虫-Selenium库

- id定位

  标签的 id 具有唯一性，就像人的身份证，我们可以通过 id 定位到它，由于 id 的唯一性，我们可以不用管其他的标签的内容。

```
driver.find_element_by_id(“*****")
```

- Class定位

  指定标签的类名，在页面中可以不唯一

```
driver.find_element_by_name(“*****")
```

- Xpath定位

  在 XML 文档中定位元素的语言，它拥有多种定位方式

```
driver.find_element_by_xpath(“*****")
```

- CSS定位

  使用选择器来为页面元素绑定属性，它可以较为灵活的选择控件的任意属性，一般定位速度比 xpath 要快，但使用起来略有难度

网络爬虫-Selenium库

- CSS定位

使用选择器来为页面元素绑定属性，它可以较为灵活的选择控件的任意属性，一般定位速度比 xpath 要快，但使用起来略有难度

方法	例子	描述
.class	`.toolbar-search-container`	选择 `class = 'toolbar-search-container'` 的所有元素
#id	`#toolbar-search-input`	选择 `id = 'toolbar-search-input'` 的元素
*	`*`	选择所有元素
element	`input`	选择所有 `<input\>` 元素
element>element	`div>input`	选择父元素为 `<div\>` 的所有 `<input\>` 元素
element+element	`div+input`	选择同一级中在 `<div\>` 之后的所有 `<input\>` 元素
[attribute=value]	`type='text'`	选择 `type = 'text'` 的所有元素

```
driver.find_element_by_css_selector('*****')
```

网络爬虫-Selenium库

```
>>> from selenium import webdriver
>>> service = Service(executable_path='chromedriver.exe’)
>>> # chrome 浏览器
>>> browser = webdriver.Chrome(service=service)
>>> # 设置浏览器浏览器的宽高为：600x800
>>> browser.set_window_size(600, 800)
>>> #浏览器全屏显示
>>> driver.maximize_window()
```

网络爬虫-Selenium库

```
>>> browser.get('https://www.csdn.net/’）
>>> # 访问CSDN个人主页
>>> browser.get('https://blog.csdn.net/qq_43965708’）
>>> #返回（后退）到CSDN首页
>>> browser.back()
>>> #前进到个人主页
>>> browser.forward()
```

网络爬虫-Selenium库

```
>>> # 在原页面打开
>>> browser.get('https://blog.csdn.net/qq_43965708’)
>>> # 新标签中打开
>>> js = "window.open('https://blog.csdn.net/qq_43965708’)”
>>> browser.execute_script(js)
```

在很多时候我们都需要用到窗口切换，比如：当我们点击注册按钮时，它一般会打开一个新的标签页，但实际上代码并没有切换到最新页面中，这时你如果要定位注册页面的标签就会发现定位不到，这时就需要将实际窗口切换到最新打开的那个窗口。

- `>>> # 获取打开的多个窗口句柄`
- `>>> windows = driver.window_handles`
- `>>> # 切换到当前最新打开的窗口`
- `>>> driver.switch_to.window(windows[-1])`

网络爬虫-Selenium库

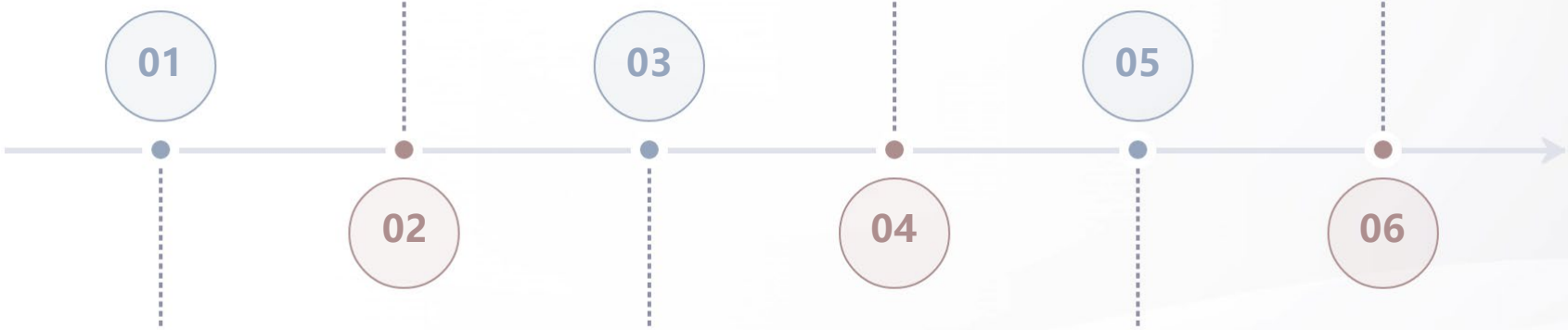
方法	描述
`send_keys()`	模拟输入指定内容
`clear()`	清除文本内容
`is_displayed()`	判断该元素是否可见
`get_attribute()`	获取标签属性值
`size`	返回元素的尺寸
`text`	返回元素文本

方法	描述
`click()`	单击左键
`context_click()`	单击右键
`double_click()`	双击
`drag_and_drop()`	拖动
`move_to_element()`	鼠标悬停
`perform()`	执行所有ActionChains中存储的动作

网络爬虫-Selenium库

```
>>> from selenium.webdriver.common.keys import Keys
>>> # 定位输入框并输入文本
>>> driver.find_element_by_id('xxx').send_keys('Dream丶killer')
>>> # 模拟回车键进行跳转（输入内容后）
>>> driver.find_element_by_id('xxx').send_keys(Keys.ENTER)
>>> # 使用 Backspace 来删除一个字符
>>> driver.find_element_by_id('xxx').send_keys(Keys.BACK_SPACE)
>>> # Ctrl + A 全选输入框中内容
>>> driver.find_element_by_id('xxx').send_keys(Keys.CONTROL, 'a')
>>> # Ctrl + C 复制输入框中内容
>>> driver.find_element_by_id('xxx').send_keys(Keys.CONTROL, 'c')
>>> # Ctrl + V 粘贴输入框中内容
driver.find_element_by_id('xxx').send_keys(Keys.CONTROL, 'v')
```

数据处理的流程



多渠道数据收集

从社交媒体、传感器、交易记录等多渠道获取原始数据，为后续分析奠定基础。

数据清洗

去除错误、不完整、格式不正确或重复的数据，确保数据质量。

提高分析准确性

通过数据清洗，提高数据分析的准确性和可靠性。

数据转换

将数据标准化、编码分类变量，便于模型理解和处理。

数据集成

合并来自不同源的数据，解决数据冲突，创建统一视图。

支持全面分析

数据集成支持更全面的分析视角，提供更深入的洞察。

数据处理

检查数据

检查一下我们刚刚读入数据的基本结构，Pandas 提供了 head() 方法打印输出前五行数据。目的是让我们对读入的数据有一个大致的了解。

```
data.head()
```

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN

5 rows × 28 columns

数据处理

检查数据

Pandas 提供了一些选择的方法，这些选择的方法可以把数据切片，也可以把数据切块。下面我们简单介绍一下：

- 查看一些基本统计信息：data.describe()

```
data.describe()
```

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	gross
count	4993.000000	5028.000000	4939.000000	5020.000000	5036.000000	4.159000e+03
mean	140.194272	107.201074	686.509212	645.009761	6560.047061	4.846841e+07
std	121.601675	25.197441	2813.328607	1665.041728	15020.759120	6.845299e+07
min	1.000000	7.000000	0.000000	0.000000	0.000000	1.620000e+02
25%	50.000000	93.000000	7.000000	133.000000	614.000000	5.340988e+06
50%	110.000000	103.000000	49.000000	371.500000	988.000000	2.551750e+07
75%	195.000000	118.000000	194.500000	636.000000	11000.000000	6.230944e+07
max	813.000000	511.000000	23000.000000	23000.000000	640000.000000	7.605058e+08

检查数据

Pandas 提供了一些选择的方法，这些选择的方法可以把数据切片，也可以把数据切块。下面我们简单介绍一下：

- 查看一列的一些基本统计信息：data.columnname.describe()
- 选择一列：data['columnname']
- 选择一列的前几行数据：data['columnsname'][:n]
- 选择多列：data[['column1','column2']]
- Where 条件过滤：data[data['columnname'] > condition]

数据处理

数据质量的分析

数据质量分析的主要任务是检查原始数据中是否存在脏数据，脏数据一般是指不符合要求，以及不能直接进行相应分析的数据。在常见的数据挖掘工作中，脏数据包括如下内容：

- 缺失值
- 异常值
- 不一致的值
- 重复数据以及含有特殊符号（如 #、 ￥、 *）的 数据

数据处理

•检查确实值：data.isnull()

```
data.isnull()
```

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	ac
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	True	False	True	True	False	True	False	
...	...	...	...	...	...	...	...	
5038	False	False	False	False	False	False	False	
5039	False	True	False	False	True	False	False	
5040	False	False	False	False	False	False	False	
5041	False	False	False	False	False	False	False	
5042	False	False	False	False	False	False	False	

数据处理

产生缺失值的原因：

a.有的信息暂时无法获取，或者获取信息的代价太大

b.有些信息是被遗漏的。可能是因为输入时认为不重要、忘记填写或对数据理解错误等一些人为因素而遗漏，也可能是由于数据采集设备的故障、存储介质的故障、传输媒体的故障等非人为原因而丢失。

c.属性值不存在。在某些情况下，缺失值并不意味着数据有错误。对一些对象来说某些属性值是不存在的，如一个未婚者的配偶姓名、一个儿童的固定收入等。

数据处理

无论什么原因，只要有空白值得存在，就会引起后续的数据分析的错误。下面介绍几个处理缺失数据的方法：

为缺失数据赋值默认值

```
data.country= data.country.fillna('')
```

对于我们的例子，我们检查一下“country”列。这一列非常简单，然而有一些电影没有提供地区，所以有些数据的值是
NaN。在我们的案例中，我们推断地区并不是很重要，所以，我们可是使用“”空字符串或其他默认值。

```
data.duration = data.duration.fillna(data.duration.mean())
```

使用数字类型的数据，比如，电影的时长，计算像电影平均时长（或者中位数等）可以帮我们。

这并不是最优解，但这个持续时间是根据其他数据估算出来的。这样的方式下，就不会因为像 0 或者 NaN这样的值在我们分析的时候而抛错。

数据处理

去掉/删除缺失数据行

假设我们想删除任何有缺失值得行。这种操作太据侵略性，但是我们可以根据我们的需要进行扩展。

```
data.dropna()
```

当然，我们也可以删除一整行的值都为 NA：

```
data.dropna(how='all')
```

我们也可以增加一些限制，在一行中有多少非空值的数据是可以保留下来的（在下面的例子中，行数据中至少要有 5 个非空值）

```
data.drop(thresh=5)
```

数据处理

去掉/删除缺失率高的列

我们可以上面的操作应用到列上。我们仅仅需要在代码上使用 axis=1 参数。这个意思就是操作列而不是行。（我们已经在行的例子中使用了 axis=0，因为如果我们不传参数 axis，默认是axis=0。）

删除一正列为 NA 的列：

```
data.drop(axis=1,how='all')
```

数据处理-规范化数据类型

有的时候，尤其当我们读取 csv 中一串数字的时候，有的时候数值类型的数字被读成字符串的数字，或将字符串的数字读成数据值类型的数字。Pandas 还是提供了规范化我们数据类型的方式：

```
data = pd.read_csv('movie_metadata.csv', dtype={'duration': int})
```

这就是告诉 Pandas ‘duration’列的类型是数值类型。同样的，如果想把上映年读成字符串而不是数值类型，我们使用和上面类似的方法：

```
data = pd.read_csv('./data/moive_metadata.csv', dtype={'title_year':str})
```

数据处理-必要的变换

人工录入的数据可能都需要进行一些必要的变换。

- 错别字
- 英文单词时大小写的不统一
- 输入了额外的空格

将我们数据中所有的 movie_title 改成大写：

```
data['movie_title'].str.upper()
```

同样的，干掉末尾空格：

```
data['movie_title'].str.strip()
```

数据处理-重复数据

首先我们校验一下是否存在重复记录。如果存在重复记录，就使用 Pandas 提供的 drop_duplicates() 来删除重复数据。

```
# 删除重复数据行
df.drop_duplicates(['first_name','last_name'],inplace=True)
```

数据处理-异常值

异常值是指样本中的个别值，其数值明显偏离其余的观测值。异常值也称为离群点，异常值的分析也称为离群点分析。

a. 简单统计量分析：

可以先对变量做一个描述性统计，进而查看哪些数据是不合理的。最常用的统计量是最大值和最小值，用来判断这个变量的取值是否超出了合理的范围。如**客户年龄的最大值为199岁，则该变量的取值存在异常。**

数据处理-异常值

b. 3σ 原则：

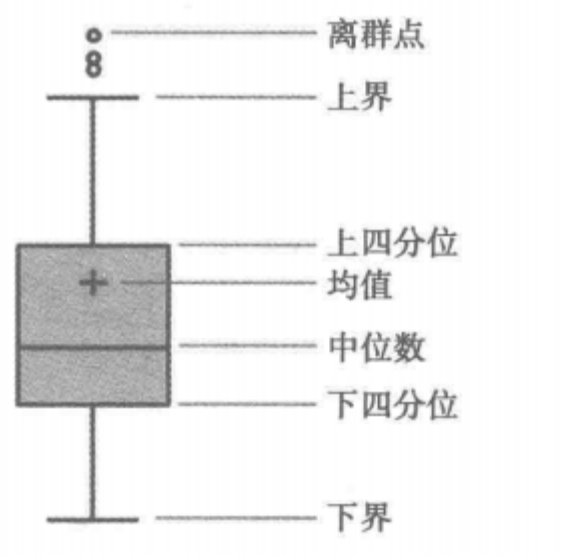
如果数据服从正态分布，在3σ 原则下，异常值被定义为一组测定值中与平均值的偏差超过3倍标准差的值。在正态分布的假设下，距离平均值3σ 之外的值出现的概率为P(|x-μ| >3σ)≤0.003，属于极个别的小概率事件。如果数据不服从正态分布，也可以用远离平均值的多少倍标准差来描述。

```
# 过滤掉 大于3陪标准差的行
# 标准差 df.std()，绝对值 df.abs()
# cond：找到每一个元素是否大于3陪标准差
cond = df.abs() > df.std()*3
cond
```

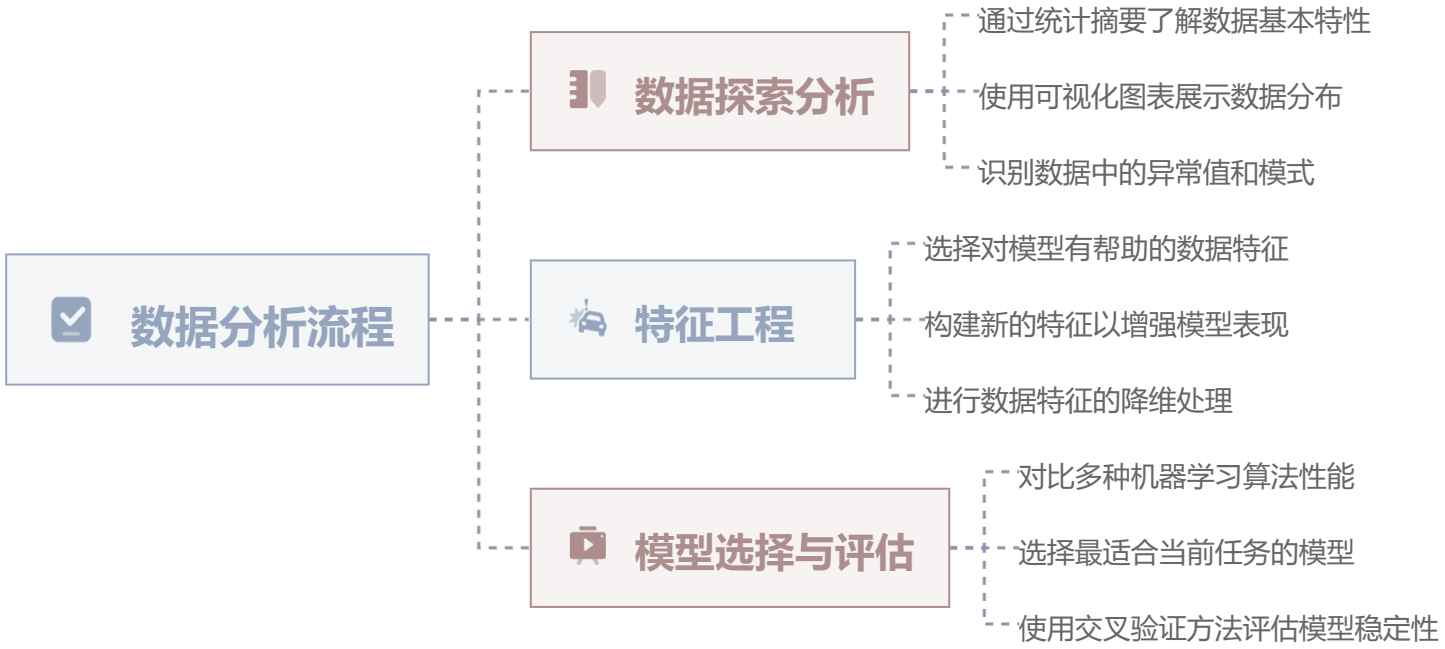
数据处理-异常值

箱型图分析

箱型图依据实际数据绘制，没有对数据作任何限制性要求(如服从某种特定的分布形式)，它只是真实直观地表现数据分布i的本来面貌；另一方面，箱型图判断异常值的标准以四分位数和四分位距为础，四分位数具有一定的鲁棒性：多达25%的数据可以变得任意远而不会很大地扰动四分位数，所以异常值不能对这个标准施加影响。**由此可见，箱型图识别异常值的结果比较客观，在识别异常值方面有一定的优越性。**



数据分析流程


数据分析流程
数据探索分析
通过统计摘要了解数据基本特性
使用可视化图表展示数据分布
识别数据中的异常值和模式
特征工程
选择对模型有帮助的数据特征
构建新的特征以增强模型表现
进行数据特征的降维处理
模型选择与评估
对比多种机器学习算法性能
选择最适合当前任务的模型
使用交叉验证方法评估模型稳定性

数据分析流程

交由后面的PPT