Python 综合面试题

第一部分 Python 基础

1. 如何理解 python 中的深度拷贝和浅拷贝

浅拷贝旨在减少内存的占用，深拷贝可以在做数据的清洗、修改或者入库的时候，对原数据进行复制一份，以防数据修改之后，找不到原数据。深浅拷贝对于可变和不可变类型的数据不同。

不可变类型包括数值类型、字符串、布尔、None、元组

可变类型：列表、字典

对于不可变对象类型，没有深浅拷贝的说法，无论是深拷贝还是浅拷贝结果一样的，如果对其重新赋值，也只是新创建一个对象，替换掉旧的而已。

对于可变类型来说，浅拷贝只复制容器，不复制容器中元素；深拷贝复制容器，元素如果是可变类型，也复制元素

2. 请说说 get 请求与 post 请求的不同:

1.GET 表示从指定的服务器中获取数据 ,POST 表示提交数据给指定的服务器处理

2.GET 查询的字符串参数显示在地址栏的 URL 中，请求参数是可见的.POST 查询字符串不会显示在地址栏中,请求参数是不可见的

3.GET 请求能够被缓存,POST 请求不能被缓存下来

4.GET 请求会保存在浏览器的浏览记录中,POST 请求不会保存在浏览器浏览记录中

5.GET 请求有长度限制,POST 请求没有长度限制

6.GET 查询的字符串参数会显示在地址栏的 URL 中，不安全,请不要使用 GET 请求提交敏感数据

POST 请求数据不会显示在地址栏中，也不会缓存下来或保存在浏览记录中，所以 POST 请求比 GET 请求安全，但也不是最安全的方式。如需要传送敏感数据，请使用加密方式传输

3. python2 和 python3 的区别

第一、Python2 中创建类区分新式类和旧式类，默认创建旧式类，继承 object 则创建新式类。Python3 中则移除了旧式类，默认创建新式类。

第二、python2 中字节字符串类型为 str，文本字符串类型为 Unicode python3 中字节字符串类型为 bytes，文本字符串类型为 str

第三、python2 中直接赋值即可声明元类，python3 中则通过 class 关键字声明元类

第四、一些 python2 中的类库，在 python3 中进行了合并，如 StringIO 和 cStringIO 合并为 IO 等

第五、python2 中默认的字符串类型默认是 ASCII 编码，python3 中默认的字符串类型是 Unicode 编码，无论字节数占用大小，均占用字符串的一个下标。

第六、python2 中，print 是个特殊语句，python3 中 print 是函数

第七、python2 中单斜杠除法运算的结果是整型，python3 中是浮点类型，如 3 通过单斜杠除号除以 2，在 python2 中等于 1，在 python3 中等于 1.5

第八、python2 中异常链会丢失原始异常信息，python3 中可以显式指定一个异常作为另一个异常的子句

第九、python2 中的 dict 类中的 keys、values 和 items 属性均返回 list 对象，iterkeys、itervalues 和 iteritems 返回生成器对象。

python3 中移除了 list、只返回一个生成器的对象。

第十、python3 中需要使用 from 进行相对导入，python2 中仅要求不写路径即可

4. Python 中 classmethod 与 staticmethod 的区别

classmethod 是类方法，staticmethod 是静态方法，都属于类，为所有对象共有；但 classmethod 有参数 cls，这个参数是类对象，由系统自动传递；staticmethod 没有类对象参数。

classmethod 在一些工厂类的情况下使用较多，也就是说 OOP 里继承的时候使用，staticmethod 一般情况下可以替换为外部的函数，后者继承的时候不可更改，和 C++/JAVA 中的静态方法很相似

5. 如何在不改变原有 list 顺序的前提下,去除该 list 中的重复代码

1, 使用循环实现

2, 使用列表生成式

3, 使用排序实现

6. 请说说,python 中不同类型的数据变量在内存中是如何存储的

python 中一切皆对象，对象的存储，采用了引用语义的方式；系统会保存对象的引用数量，当某个对象的引用数量为 0 时，对象会被回收。

变量定义： a = 3 在这里 3 就是一个整型对象，3 这个对象的地址保存到变量 a 中，变量存储的只是一个对象的内存地址，而不是这个对象本身

变量保存了对象的地址，也称变量引用了对象

7. 请解释 with 关键字的用法

上下文管理器：上下文管理器对象实现了__enter__() 和 __exit__() 方法。__enter__() 方法在语句体执行之前进入上下文时执行 ，__exit__() 在语句体执行完后从上下文退出时执行。

with 语句支持运行时上下文这一概念。

语法：with 表达式 [as 变量]:

语句体

作用：经常用于保存和恢复各种全局状态，锁定和解锁资源，关闭打开的文件等。

例子：**with open('1.txt') as fp:**

for line in fp:

prin(line)

这里使用了 with 语句，不管在处理文件过程中是否发生异常，都能保证 with 语句执行完毕后已经关闭了打开的文件句柄。如果不用 with 语句，则要使用 try-except-finally 处理，语法变的复杂。

8. 请说说 python 中正则表达式如何提取数据

利用正则表达式来提取想要的内容要使用分组。

分组就是用一对圆括号“()”括起来的正则表达式，匹配出的内容就表示一个分组。从正则表达式的左边开始看，看到的第一个左括号“(”表示第一个分组，第二个表示第二个分组，依次类推，需要注意的是，有一个隐含的全局分组（就是 0），就是整个正则表达式。

要想提取数据，首先你需要写出正确的正则表达式，在表达式中把要提取的内容使用分组括起来；然后执行 match 或 search 匹配；匹配后使用 group（num）和 groups 方法获取提取内容，也可以使用 findall()

9. 请说说 python 中 list,set,dict,tuple 的区别

list 和 tuple 区别：

1）list 用[]定义，tuple 用（ ）定义

2）list 是可变容器，可以修改其中的元素，tuple 属于不可变容器，不能修改元素。

3）tuple 属于不可变容器，没有深复制和浅复制，list 有深复制和浅复制

4）list 一般不做函数参数的默认值，tuple 可以做函数参数的默认值

dict 和 set 区别：

1）dict 可以通过 key 访问，set 不行

2）dict 键不可以重复，值可以重复，set 不允许有重复元素

3）dict 可以根据键修改值，set 不能修改元素

10.请说说队列与栈两种数据结构如何实现

栈：后进先出，可以使用列表实现，只需要把操作固定到列表一端，这个列表就可以当栈使用

队列：先进先出，可以使用列表实现，在一端固定插入元素，在另外一端删除元素就可以实现队

11.请说说 python 中*args 与**kwargs 的用法与区别

python 的函数参数可以有多种形式，其中如果要接收任意多个位置参数，可以将参数声明为*args，其中 args 只是变量名字，可以是任何合法变量名，例如函数 def demo(*args)；可以接收以下传参：

```
demo(1)
demo(1,2)
demo(1,2,3,4)
```

如果函数形参声明为**kwargs，则表示函数可以接收任意关键字参数，比如 def test(**kwargs)，则可以有如下调用方式：

```
test(a=1)
test(a=1,b=3)
test(a=1,b=5,d=100)
```

区别：

1）*args 中 args 是元组，可以通过 args 元组获取所有位置参数

2）**kwargs 中的 kwargs 是字典，可以通过 kwargs 获取所有关键字参数.

3）在函数的形参中，如果同时有*args 和**kwargs，*args 必须在**kwargs 前面.

12.请说说 python 中 type 函数与 isinstance() 的区别

1, 目的不同: type 主要用于返回对象的类型，isinstance 用于判断对象是否是指定类型

2, 返回值不同: type 返回类型，isinstance 返回 True 或 False

3, type 不能判断子类对象是否属于父类，而 isinstance 可以

13.请叙述下 python 如何复制一个大文件

1）打开源文件（读）和目标文件（写）

2）从源文件中读取指定大小的字节，比如 1024，将其写入目标文件

3）重复 2）直到源文件读完

4）关闭源文件和目标文件

```
def mycopy(src_file, dst_file):
    try:
        with open(src_file, "rb") as fr, open(dst_file, 'wb') as fw:
            while True:
                data = fr.read(1024)
                if not data:
                    break
                fw.write(data)
    except OSError:
        print("打开读文件失败")
    except:
        print("可能 U 盘被拔出...")
```

14.请解释下什么是解释性语言,什么是编译性语言

编译型语言：编译型语言写的程序执行之前，需要一个专门的编译过程，把程序编译成机器语言文件；比如，exe 文件，以后运行的话就不用重新编译了，直接运行；因为翻译只做了一次，运行时不需要编译，所以编译型语言的程序执行效率高！

解释型语言：解释型语言的程序不需要编译，省了道工序，解释型语言在运行程序的时候才翻译，比如解释型 basic 语言，专门有一个解释器能够直接执行 basic 程序，每个语句都是执行的时候才翻译；这样解释型语言每执行一次就要翻译一次，效率比较低。但是解释型语言跨平台性比较好，因为他依赖解释器，只要平台装有解释器程序就行。

15.请简单的叙述下数据存储原理

（1）顺序存储方法

该方法把逻辑上相邻的结点存储在物理位置上相邻的存储单元里，结点间的逻辑关系由存储单元的邻接关系来体现。由此得到的存储表示称为顺序存储结构 （Sequential Storage Structure ），通常借助程序语言的数组描述。该方法主要应用于线性的数据结构。非线性的数据结构也可通过某种线性化的方法实现顺序存储。

（2）链接存储方法

该方法不要求逻辑上相邻的结点在物理位置上亦相邻，结点间的逻辑关系由附加的指针字段表示。由此得到的存储表示称为链式存储结构（Linked Storage Structure），通常借助于程序语言的指针类型描述。

（3）索引存储方法

该方法通常在储存结点信息的同时，还建立附加的索引表。索引表由若干索引项组成。索引项的一般形式是：(关键字、地址)。关键字是能唯一标识一个结点的那些数据项。

（4）散列存储方法

该方法的基本思想是：根据结点的关键字直接计算出该结点的存储地址。

16.请说下您平时是怎么调错的

1）print 调错：把所有可能出错的变量都打印出来，看看是哪段代码出错。用 print 最大的坏处是将来还得删掉它

2）assert 断言调试：凡是用 print 来辅助查看的地方，都可以用断言（assert）来替代。如果断言失败，assert 语句本身就会抛出 AssertionError，不过，启动 Python 解释器时可以用-O 参数来关闭 assert。关闭后，你可以把所有的 assert 语句当成 pass 来看。

3）可以启动 Python 的调试器 pdb 让程序以单步方式运行，可以随时查看运行状态。

4）使用 IDE 调试：可以使用 pycharm、pydev 等集成开发环境设置断点，单步调试。

17.请说说 python 中 while-else/for-else 语法的用法

while-else：

```
while 表达式：
    循环体
else:
    【else 语句块】
```

说明：当 while 语句执行完成之后，执行【else 语句块】，如果用 break 跳出循环则不执行 else

for-else:

```
for x in 表达式：
    循环体
else:
    【else 语句块】
```

说明：当 for 语句执行完成之后，执行【else 语句块】，如果用 break 跳出循环则不执行 else

18.请说说 python 中的作用域是怎样划分的

概念：变量的作用域指的是变量在那段代码中可以使用，可以使用变量的那段代码就是变量的作用域。在 python 中，只有函数/类/模块才引入作用域，if/elif/else，while/for,try/except 等并不会引入新的作用域

作用域按从小到大顺序可以分为：

局部作用域：在函数内部定义，从定义开始到本函数结束

闭包作用域：外部函数定义的变量，范围从定义开始到外部函数结束，可以在闭包函数内直接使用，在整个程序运行过程中都存在

全局作用域：在所有函数外定义，范围从定义开始到本文件结束

内建作用域：python 系统内定义的函数、类等，可以在任何位置直接使用

19.请说说你是如何理解 python 中的封装,继承,多态的

封装：封装就是把对象的属性和行为结合成一个独立的整体，把内部的实现细节隐藏起来，不能被外界所看见，调用的人只能执行，而看不到实现的细节，保证了代码的安全性

继承：从已有类中派生出新的实例对象，子类继承父类的属性和方法，并且自己可以拓展出自己独有的属性和方法。保证了代码的复用率。python 支持单继承和多继承。

多态：首先 Python 不支持多态，也不用支持多态，python 是一种多态语言，崇尚鸭子类型。在鸭子类型中，关注的不是对象的类型本身，而是它是如何使用的。例如，在不使用鸭子类型的语言中，我们可以编写一个函数，它接受一个类型为鸭的对象，并调用它的走和叫方法。在使用鸭子类型的语言中，这样的一个函数可以接受一个任意类型的对象，并调用它的走和叫方法。如果这些需要被调用的方法不存在，那么将引发一个运行时错误。任何拥有这样的正确的走和叫方法的对象都可被函数接受的这种行为引出了以上表述，这种决定类型的方式因此得名。

20.请从 socket 编程的角度谈谈网络请求的流程

网络请求是基于 HTTP 协议的，但 http 协议只是一个应用层协议，它底层是通过 TCP 进行传输数据的。因此，浏览器访问 Web 服务器的过程必须先有“连接建立”的发生。浏览器向 Web 服务器发出 Http 请求以及 Web 服务器给浏览器回复的过程如下：

1）浏览器创建 Socket，按给定 IP（域名）和端口（默认为 80）连接服

务器。

2）连接成功后，浏览器依据 HTTP 协议规范，向 Web 服务器发送请求数据，会使用类似 Socket.Send()、Socket.BeginSend()等方法。

3）浏览器等待服务器处理并返回数据；

4）Web 服务器端使用 Socket.Accept()、Socket.BeginAccept()等方法侦听到浏览器的连接后，便开始接收浏览器发送的数据。接收到请求数据后，依据 HTTP 协议规范解析数据，然后处理，最终将处理结果发回给浏览器，这里可能用到类似 Socket.Send()、Socket.BeginSend()等方法；

5）Web 服务器发送完处理结果后，关闭 Socket；

6）浏览器接收 Web 服务器发回的数据（如 html），将其显示在浏览器 UI 界面。关闭 socket；

7）一次“浏览器到 Web 服务器”的 http 请求结束；

21.请说说 TCP 与 UDP 编程的区别

1.TCP 是面向连接的，即每次收发数据之前必须通过 connect 建立连接，也是双向的，即任何一方都可以收发数据，协议本身提供了一些保障机制保证它是可靠的、有序的，即每个包按照发送的顺序到达接收方。

UDP 它是无连接的，不可靠的，因为通讯双方发送数据后不知道对方是否已经收到数据，是否正常收到数据。

任何一方建立一个 socket 以后就可以用 sendto 发送数据，也可以用 recvfrom 接收数据。根本不关心对方是否存在，是否发送了数据。

它的特点是通讯速度比较快。大家都知道 TCP 是要经过三次握手的，而 UDP 没有。

2.TCP 对系统资源的要求较多，UDP 相对较少

3.TCP 程序结构复杂,UDP 程序结构较简单

4.TCP 以流模式传输数据与,UDP 以数据报模式传输数据

5.TCP 能够保证数据正确性，UDP 则可能丢包

6.TCP 能过保证数据顺序，UDP 则不保证

22.请说说 python 中高阶函数的思想

请说说你对 python 中高阶函数的认识:

高阶函数指的是能接收函数作为参数的函数；python 中有一些内置的高阶函数，在某些场合使用可以提高代码的效率

1. map 函数：接收一个函数和一个序列，map 将传入的函数依次作用到序列的每个元素，并且把结果作为新的列表返回.所以 map 函数可以方便将一

个序列中的每一个元素做统一处理.

2.reduce 函数：接收一个函数和一个序列,把一个函数作用到一个序列上，这个函数必须接收两个参数，reduce 把结果和序列的下一个元素做累积计算.所以 reduce 函数可以方便将一个序列中的所有元素做累计处理.

3.filter 函数：接收一个函数和一个序列，把传入的函数依次作用于每个元素,然后根据返回值是 True 还是 False 决定保留还是丢弃该元素. 所以 filter 函数可以方便的过滤掉序列的无用元素.

23.请说说迭代器与可迭代对象的区别及迭代器的好处

请说说迭代器与可迭代对象的区别及迭代器的优点:

1.可迭代对象: 可迭代对象并不是一种具体的数据类型,只要可以直接用于 for…in…循环遍历的对象都是可迭代对象，比如：list，tuple，dict，set，str 等等。

2.迭代器:可以被 next()函数调用并不断返回下一个值的对象称为迭代器：Iterator。它们表示一个惰性计算的序列；

迭代器优点:使用迭代器不要求事先准备好整个迭代过程中的所有元素。迭代器仅仅在迭代到某个元素时才计算该元素，

而在这之前或之后元素可以不存在或者被销毁。比起使用列表等容器存储，更加省内存,因此迭代器适合遍历一些数量巨大甚至无限的序列。

24.请说说装饰器的好处,什么情况下使用,以及如何实现一个通用装饰器

1.装饰器的实现是由闭包支撑的；

2.装饰器本质上是一个 python 函数，它可以在让其他函数在不需要做任何代码的变动的前提下增加额外的功能；

3.装饰器的返回值也是一个函数的对象，该函数对象就是添加额外功能后的与原函数同名的函数.

通用装饰器:

```
# 装饰器; 在不修改源代码的基础上,添加新的功能
def outer(functionName):
    def inner(*args, **kwargs):
        print("-----添加功能-----")
        ret = functionName(*args, **kwargs)
        print("-----添加功能-----")
        return ret
    return inner
```

25.请说说线程什么情况下需要加锁，及其优缺点

加锁原因：如果有多个线程同时操作一个对象，如果没有很好地保护该对象，会造成程序结果的不可预期

比如:我们在每个线程的 run 方法中加入一个 time.sleep(1)，并同时输出线程名称，则我们会发现，输出会乱七八糟。

因为可能我们的一个 print 语句只打印出一半的字符，这个线程就被暂停，执行另一个去了，所以我们看到的结果很乱，这种现象叫做“线程不安全”

解决:

Threading 模块为我们提供了一个类 Threading.Lock 锁。我们创建一个该类对象，在线程函数执行前，“抢占”该锁，执行完成后，“释放”该锁， 则我们确保了每次只有一个线程占有该锁。这时候对一个公共的对象进行操作，则不会发生线程不安全的现象了。

锁的优点：使得数据能安全的进行操作

锁的缺点:

1.如果多个线程要调用多个对象，而 A 线程调用 A 锁占用了 A 对象，B 线程调用了 B 锁占用了 B 对象,A 线程不能调用 B 对象，B 线程不能调用 A 对象，于是一直等待。这就造成了线程“死锁”

2.如果多个线程要调用同一个对象，A 线程调用 A 锁占用了 A 对象，B 线程也需要使用 A 对象,A 线程不释放 A 对象，B 线程就不能调用 A 对象,B 线程会等待 A 线程释放后才能使用,所以效率低下

26.请谈谈 python 中的协程的运用

1.协程就是进程和线程的升级版,进程和线程都面临着内核态和用户态的切换问题而耗费许多切换时间，而协程就是用户自己控制切换的时机,不再需要陷入系统的内核态。

2.协程的执行效率非常高。因为子程序切换不是线程切换，而是由程序自身控制,因此，没有线程切换的开销.和多线程相比， 线程数量越多相同数量的协程体现出的优势越明显

3.协程不需要多线程的锁机制。由于只有一个线程，也不存在同时写变量的冲突，在协程中控制共享资源不需要加锁，只需要判断数据的状态， 所以执行效率远高于线程，对于多核 CPU 可以使用多进程+协程来尽可能高效率地利用 CPU

27.请说说计算机中使用缓存的好处

从系统的层面说，CPU 的速度远远高于磁盘 IO 的速度；所以要想提高响应速度，必须减少磁盘 IO 的操作，但是有很多信息又是存在数据库当中的，每次查询数据库就是一次 IO 操作；要想提高系统的性能，尽量减少 IO 的操作，特别是磁盘 IO 的操作；使用缓存可以有效的避免这种情况；所以在架构设计过程中，涉及到查询数据库的时候，应该考虑一下是不是考虑使用缓存技术来提高系统的性能，并且降低数据库的负载。

28.请说说冒泡,选择,插入,快速排序

需要掌握这几个排序算法

思考下面问题:

1. Python 中如何使用多核.
2. 请详细描述你对 python 中的多线程的理解,什么情况下使用多线程.
3. 谈谈你对面向对象思想的理解，为什么要使用面向对象.
4. 谈谈你对分布式的理解
5. 使用进程池和线程池的好处
6. 谈谈并发和并行的区别，以及什么情况下使用
7. 请说说生成器的作用，什么情况下使用?
8. 请谈谈你对 Python 闭包的理解
9. Python 中__new__()和__init()__的区别

第二部分 Web 前端&Linux&SQL

1. 什么是 Ajax，常用场景有哪些

   ajax 的全称是 Asynchronous Javascript And XML， 即为异步 JS 和 XML，涉及到的核心类有 XMLHttpRequest 和 FormData。主要用于网页中异步请求服务器的资源和局部刷新网页中的数据；交互数据的格式常用 xml、json 两种。

2. 简述 xml 和 html 的关系

   两者都是标记语言，HTML 一个是超文本标记语言，XML 一个是扩展标记语言。区别主要三个方面：（1）HTML 不具备扩展性，而 XML 是原标记语言，可以用于定义新的标记语言;（2）HTML 侧重于如何表现，而 XML 是侧重于如何描述信息;（3）对于标记的嵌套和配对，HTML 较松; XML 则是严格要求，它是遵循 DTD 的树形结构;

   总而言之 HTML 用于网页数据展示(表格、图片、视频等)，XML 用于数据交互。

3. 简述一下 CSS 盒子模型

   CSS 盒子模型又叫框模型，包括内容(content)、填充(padding)、边框(border)、边界(margin)等属性。

   border:元素的边框，用于将框的边缘与其他框分开；

   margin：外边距，表示框的边缘与相邻框之间的距离，也称为页边空白；

   padding:内边距，表示框内容和边框之间的空间;

   另外，width 和 height 指的是内容区域的宽度和高度。

   增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。

   而外边距则可以在多个元素框之间创建空白，避免这些框都挤在一起。总之用了 padding 属性或者 margin 属性设置了元素的边距以后，会增加元素在页面布局中所占的面积。

4. 简述一下 src 与 href 的区别

   href 是 Hypertext Reference 的缩写，表示超文本引用。用来建立当前元素和文档之间的链接。常用于的标签有：<link>、<a>。src 是 source 的缩写，src 的内容是页面必不可少的一部分。src 指向的内容会嵌入到文档中当前标签所在的位置。常用的标签有：<img>、<script>、<iframe>、

<source>、<video>、<audio>。

5. 简述 xml 和 json 的区别

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式，XML 相比较 json 来说，XML 文件较大，格式复杂，解析比较困难。而 json 可以转成 js 对象，解析简单，传速快; json 是目前用于前后端交换的常用数据格式。

6. px，em，rem 的区别

px 是像素单位，一般指定固定大小使用， 如 16px。em 和 rem 是相对单位，em 相对于父级字体大小的倍数，rem 是相对根级(root)字体大小的倍数，如 1.5em, 1.2rem。

7. CSS 中的定位模式有哪些？有何特点

static：默认值，没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。

relative：生成相对定位的元素，相对于其正常位置进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。

fixed：元素框的表现类似于将 position 设置为 absolute，不过其包含块是视窗本身。

absolute：生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。

元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。

8. 浅谈 DML、DDL、DCL 的区别

DML（data manipulation language）数据库操纵语言：就是我们最经常用到的 SELECT、UPDATE、INSERT、DELETE 等语句，主要用来对数据库的数据进行一些操作。

DDL（data definition language）数据库定义语言：其实就是我们在创建表的时候用到的一些 sql，比如说：CREATE、ALTER、DROP 等语名。DDL 主要是用在定义或改变表的结构，数据类型，表之间的链接和约束等初始化工作上。

DCL（Data Control Language）数据库控制语言：是用来设置或更改数据库用户或角色权限的语句，包括（grant,deny,revoke 等）语句。这个比较少用到。

9. JavaScript 的核心组成有哪些

javascript 主要有以下三大核心：ECMAScript 、DOM（document object model）和 BOM（browser object document ）。

（1）ECMA 是规定了 js 的语法标准。

（2）DOM 是文档对象模型，规定了文档的显示结构，可以轻松地删除、添加和替换节点

（3）BOM 是浏览器对象模型，就是浏览器自带的一些功能样式，如搜索框、设置等浏览器窗口交互的对象

10. eval 是做什么的

eval()函数把字符串参数解析成代码并运行，并将执行的结果作为 eval()的执行结果进行返回。

11. JavaScript 中的定时器有哪些？它们的区别以及用法是什么

JS 中的定时器有两种：setInterval(function, ms)和 setTimeout(function, ms), setInterval()是每隔多少毫秒执行一次 function 函数，setTimeout()是延迟多少毫秒后执行 function 函数。它们都返回一个 id,通过 clearInterval(id)函数来取消 Interval 定时器, clearTimeout()函数取消 Timeout 定时器;

12. jQuery 有哪些核心模块

核心模块有： jQuery()函数对象、Selector 选择器、DOM(文档)操作、CSS(样式)操作、事件处理、Ajax、动画效果等。

13. 关系型数据库与非关系型数据库的区别

关系型数据库主要由二维关系表组成的，包含视图、索引、函数、存储过程以及权限等，使用标准的 SQL 语句及相应引擎语法,目前常用的数据库引擎包括 MySQL、SQL Server、Oracle、DB2 等。

而非关系型数据库(简称：NoSQL)主要以 Key-Value 或文档结构组成，以高速读取、缓存为特点，而且不支持 SQL 语句，目前常用的 NoSQL 有 Redis、MongoDB 和 Hbase。

14. 数据库事务的四个特性及含义

数据库事务 transanction 的四个特性(ACID):

原子性(Atomicity): 整个事务中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。

一致性(Correspondence)： 在事务开始之前和事务结束以后，数据库的

完整性约束没有被破坏。

隔离性(Isolation)： 隔离状态执行事务，使它们好像是系统在给定时间内执行的唯一操作。如果有两个事务，运行在相同的时间内，执行 相同的功能，事务的隔离性将确保每一事务在系统中认为只有该事务在使用系统。这种属性有时称为串行化，为了防止事务操作间的混淆，必须串行化或序列化请求，使得在同一时间仅有一个请求用于同一数据。

持久性(Durability): 在事务完成以后，该事务所对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。

15.数据库索引的工作原理及种类

数据库索引本质上是一个用来协助数据库快速查询更新数据的排序的数据结构，一般情况下均采用 B 树及其变种 B+树来实现。

数据库的索引从不同的维度上我们可以将其划分为聚簇索引、非聚簇索引或者唯一索引、覆盖索引。

16.常见数据库的引擎以及应用场景

mysql 常见的存储引擎有 InnoDB、MyISAM 和 MEMORY。

InnoDB 是事务型数据库的首选引擎，支持事务安全表，支持行锁定和外键。主要应用于需要提供提交、回滚、崩溃恢复能力的事务安全能力，并要求实现并发控制的场景。

MyISAM 基于 ISAM 存储引擎，并对其进行扩展。它是在 Web、数据仓储和其他应用环境下最常使用的存储引擎之一。MyISAM 拥有较高的插入、查询速度，但不支持事务。如果数据表主要用来插入和查询记录，则 MyISAM 引擎能提供较高的处理效率。

MEMORY 存储引擎将表中的数据存储到内存中，未查询和引用其他表数据提供快速访问。只是临时存放数据，数据量不大，并且不需要较高的数据安全性，对于这种场景 MEMORY 引擎是一个不错的选择。

17.环境变量的作用？Linux 环境变量的种类

环境变量是设置给系统或者用户的应用程序的一些变量，其具体的作用和具体的环境变量相关。比如 PATH 变量就是告诉系统当系统运行一个程序而没有完整路径时将在那些目录下寻找程序。环境变量主要分为系统环境变量和用户环境变量。

18.如何理解 Linux 中的软硬链接

linux 中的硬链接直接指向文件的 inode，而删除硬链接仅仅删除链接本身，不会影响文件本身。与之相对的软连接则是在链接文件的数据中存储目标文件

的路径。使用 ln -s 创建软连接，不带-s 选项表示创建硬连接。

19.简要介绍一下 Linux 的启动流程

linux 系统启动流程可以简单总结为以下几步：

1）开机 BIOS 自检，加载硬盘。

2）读取 MBR,进行 MBR 引导。

3）加载 grub 引导菜单(Boot Loader)。

4）加载 Linux 内核（kernel）。

5）启动 init 进程，依据 inittab 文件设定运行级别。

6）init 进程，执行 rc.sysinit 文件。

7）启动内核模块，执行不同级别的脚本程序。

8）执行/etc/rc.d/rc.local。

9）启动 mingetty，进入系统登陆界面。

20.日志的作用？如何查看 Linux 启动日志

日志是记录系统中硬件、软件和系统问题的信息，同时还可以监视系统中发生的事件。用户可以通过它来检查错误发生的原因，或者寻找受到攻击时攻击者留下的痕迹。在 linux 中的启动日志一般存在/var/log/boot.log 或者 message.log 中，所以我们可以通过 tail /var/log/boot.log|message.log 来查看最新的日志记录。注意：因为日志文件可能比较大，慎用 cat 查看。

21.Linux 系统运行状态有哪些，如何切换

Linux 系统有 7 个运行状态即运行级别：0-系统停机状态、1-单用户工作状态、2-多用户状态、3-完全的多用户状态、4-系统未用(保留)，5-X11 控制台模式、6-系统正常关闭并重启。

修改运行状态可以通过修改 inittab 文件中的 initdefault 选项为制定的 id 达到修改目的。

需要注意的是：

1、系统默认运行级别不能设为 0 和 6，否则不能正常启动。

2、单用户工作状态，root 权限，用于系统维护，禁止远程登陆。

3、X11 控制台，登陆后进入图形 GUI 模式。

22.shell 是什么，有什么作用

Shell 字面理解就是个“壳”，是操作系统（内核）与用户之间的桥梁，充当命令解释器的作用，将用户输入的命令翻译给系统执行。Linux 中的 shell 与 Windows 下的 DOS 一样，提供一些内建命令（shell 命令）供用户使用，可以用这些命令编写 shell 脚本来完成复杂重复性的工作。

23.简述 Linux 中文件权限的设计

在 Linux 中，文件权限由三部分组成，每一部分由 rwx 三个字母组成。r 表示只读， w 表示只写， x 表示可执行，如果不具有某一权限，可以用“-”表示。如读写权限的写法： rw-。权限的第一部分表示用户的权限，第二部分表示组中其它用户权限，第三部分表示其他用户的权限。另外，每一部分的权限也可以用数字累加表示，rwx 对应 421，即具有 rw 权限，则表示 6，在 Linux 中可以用 chmod 命令来修改文件权限，如设置当前用户拥有 /opt/h.py 文件的读写和执行权限： sudo chmod 700 /opt/h.py

24.什么是闭包，有什么特性，简要介绍你理解的闭包

闭包是指有权访问另一个函数作用域中的变量的函数（即当函数被调用时会创建一个执行函数和相应的作用域链。作用域链本质上是指向变量对象的指针列表，只引用，不实际包含变量对象）。作用域链中，函数内部变量对象优先级最高，然后，由最近的外部函数依次向后排。

闭包具有封闭性和持久性，封闭性：外界无法访问闭包内部的数据，如果在闭包内声明变量，外界是无法访问的，除非闭包主动向外界提供访问接口；持久性：一般的函数，调用完毕之后，系统自动注销函数，而对于闭包来说，在外部函数被调用之后，闭包结构依然存在。

缺点：由于闭包携带包含其它函数的作用域，因此比其他函数占用的内存更多。

优点：减少创建全局变量 ； 减少传递给函数的参数量;

闭包可以读取函数内部的局部变量，这些变量的值始终保存;在内存中函数执行后，函数执行环境的作用域会被销毁，但是活动对象不会销毁，只有将匿名函数销毁后才可以销毁活动对象。可以将保存函数的变量赋值为 null，将可销毁匿名函数作用域。

常见创建闭包方法：在一个函数内部创建另一个函数。

思考下面问题:

1. git 的常用命令, git 提交项目的流程, git 如何解决冲突, 谈谈你是如何使用 git 分支的
2. 前端获取后端数据有哪些方式
3. 请谈谈你对 Http 协议的理解
4. 数据库优化的思路
5. 写一条 SQL 语句查找 Person 表中 age 第二大的人

第三部分 Django&Flask

1. 请简要概述 MVC 和 MTV

MVC 是一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码；M 模型，数据存取；V 视图，数据展示； C 控制器，业务逻辑处理。

MTV 本质上 MTV 与 MVC 模式没有什么区别，也是各组件之间为了保持松耦合关系，只是定义上有些不同；M 模型，数据存取； T 模板，数据展示； V 视图，业务逻辑处理。

2. Django 请求的生命周期

客户端发起请求 > nginx > uwsgi > 中间件 > url 路由 > view 视图 > ORM 获取数据 > 模板渲染 > 服务器返回响应 > 中间件 > uwsgi > nginx > 客户端显示

3. 简述 FBV，CBV

FBV: 用户发送 url 请求，Django 会依次遍历路由映射表中的所有记录，一旦路由映射表其中的一条匹配成功，就执行视图函数中对应的函数名；

CBV: 用户发起 url 请求，Django 会通过路由映射表匹配成功后会自动去找 dispatch 方法，接着 Django 会通过 dispatch 反射的方式找到类中对应的方法并执行类中的方法，执行完毕之后，会把客户端想要的数据返回给 dispatch 方法，由 dispatch 方法把数据返回给客户端。

4. Django 中间件的作用和应用场景

作用: 中间件是介于 request 与 response 处理之间的一道处理过程，能在全局上改变 django 的输入与输出。 应用场景: 用户认证系统、CSRF 保护、IP 限制、URL 访问限制、缓存...

5. Django 重定向是如何实现的？301 和 302 有什么区别

重定向: HttpResponseRedirect、redirect、reverse; 301 是永久重定向，搜索引擎在抓取新的内容的同时也将旧的网址替换为了重定向之后的网址；而 302 是临时重定向，搜索引擎会抓取新的内容而保留旧的地址；

6. Django 中 csrf 实现机制

在客户端页面上添加 csrf_token，服务器端进行验证，服务器端验证的工作通过'django.middleware.csrf.CsrfViewMiddleware'这个中间层来完成，若验证失败则 403 错误。在 django 当中防御 csrf 攻击的方式有两种:在表单当中附加 csrf_token、通过 request 请求中添加 X-CSRFToken 请求头。

7. 什么是 wsgi，uwsgi，uWSGI

wsgi: 用在 python web 框架编写的应用程序与后端服务器之间的规范; uwsgi: uWSGI 服务器实现的独有的协议; uWSGI: 一个 Web 服务器，它实现了 WSGI 协议、uwsgi、http 等协议，用于接收前端服务器转发的动态请求并处理后发给 web 应用程序；

8. Django 本身提供了 runserver，为什么不能用来部署？

Django 本身提供的 runserver 是属于开发服务器，这个开发服务器是没有经过安全测试的，另外 runserver 起来的其实是 HTTPServer 就是 Python 自带的 simple_server，而标准 Python 只能是是单线程，无法并发。对于一个成熟的站点提供服务，需要 Web 服务器 [静态数据] 和 App 服务器[动态数据]，还需要多线程并发处理能力。

9. 什么是会话技术，cookie，session 的区别, token 呢

会话技术，其实也就是为了状态的保持，延长请求的生命周期，因为 http 协议是无状态的，每次请求都是一次新的请求，不会保持之前的请求状态。Cookie 所有数据存储在客户端,支持数据过期，但是不能跨浏览器，不能跨域名，数据不安全，session 所有数据存储在服务端,相对数据安全，支持过期设置，但是要依赖于 cookie。

10. debug-toolbar 的作用

是一组可配置的面板集合，用来显示 Django Web 应用当前请求/响应的各种调试信息，进行性能调优，与系统调试

11. orm 中 db first 和 code first 的含义

db first 是基于已存在的数据库，之后再创建实体类，数据库对象与实体类的匹配关系等，也就是从一个数据库开始，然后生成实体框架和相应代码。

code first，根据需要，为你一个领域的对象创建类集合，而不是首先来设计数据库，然后来根据数据库设计你的类，Code-First APis 将会基于你的类和配置，为你创建数据库。

12. 简述 ORM，ORM 的优缺点

ORM 是通过使用描述对象和数据库之间映射的元数据，将程序中的对象自动持久化到关系数据库中

优点：

1 不需要再动手写 SQL 语句

2 对不同数据库管理系统的连接以及操作进行了抽象，程序员完全可以不考虑使用的是什么数据库以及数据库的版本

缺点：

1 ORM 的缺点是会牺牲程序的执行效率和会固定思维模式

2 对于复杂的 SQL 有心无力

3 无法完全屏蔽数据库底层细节

4 关系-对象映射的实现是以性能为代价，方便了开发，牺牲了效率。

13.谈谈你对前后端的理解.

14.谈谈你对 RESTful 规范的认识

REST 是所有 Web 应用都应该遵守的架构设计指导原则。面向资源是 REST 最明显的特征。将对资源的操作与 HTTP 方法在对应

15.Django REST Framework 框架中有哪些组件

视图、序列化、解析器、认证与权限、分页、throttle(访问频率)组件

16.Django 和 Flask 的区别

Flask

1 Flask 自由、灵活，可扩展性强，第三方库的选择面广，开发时可以结合自己最喜欢用的轮子，也能结合最流行最强大的 Python 库

2 入门简单，即便没有多少 web 开发经验，也能很快做出网站

3 非常适用于小型网站

4 非常适用于开发 web 服务的 API，开发大型网站无压力，但代码架构需要自己设计，开发成本取决于开发者的能力和经验

5 各方面性能均等于或优于 Django

Django：

1 Django 的自带 ORM 非常优秀，综合评价略高于 SQLAlchemy

2 自带的模板引擎简单好用

3 ORM 也使 Django 与关系型数据库耦合度过高

4 非常适合企业级网站的开发：快速、靠谱、稳定

5 Django 成熟、稳定、完善，但相比于 Flask

6 Django 的整体生态相对封闭

7 Python web 框架的先驱，用户多，第三方库最丰富 Flask 自由、灵活，可扩展性强，第三方库的选择面广，开发时可以结合自己最喜欢用的轮子，也能结合最流行最强大的 Python 库

17.列举 Http 请求中常见的请求方式以及应用场景

GET： 向特定资源发出请求（获取数据）

POST： 创建资资源（提交数据）

DELETE： 删除资源

PUT： 向指定资源位置上上传其最新内容

18.列举一下 Http 中常见状态码

200： 请求已成功。

302：临时重定向，HTTP1.0 的状态码

400： Bad Request，语法错误，服务器无法理解

403： Forbidden 权限不够

404：网络资源不存

405, Method Not Allowed 请求方式不被允许

500, Internal Server Error 服务器出错

502, Bad Gateway，作为网关或代理工作的服务器尝试执行请求时，从上游服务器接受到无效的响应

19.列举 Http 请求中常见的请求头

refer，location，user-agent，accept，accept-encoding，cookie，cache-control

20.Flask 框架的核心依赖组件

jinja2,Werkzeug 两个核心组件，Jinja2 是一个现代的，设计师友好的 Python 模板语言，模仿 Django 的模板。使用可选的沙盒模板执行环境，它具有快速，广泛使用和安全性，Werkzeug 不是一个框架，它是一个带有实用程序的库，可以创建自己的框架或应用程序，因此非常灵活。

21.Flask 的默认 session 处理机制

Flask 的默认 session 利用了 Werkzeug 的 SecureCookie，把信息做序列化(pickle)后编码(base64)，放到 cookie 里了。过期时间是通过 cookie 的过期时间实现的。为了防止 cookie 内容被篡改，session 会自动打上一个叫 session 的 hash 串，这个串是经过 session 内容、SECRET_KEY 计算出来的，看得出，这种设计虽然不能保证 session 里的内容不泄露，但至少防止了不被篡改。

22.flask 中的钩子函数有哪些？可以做什么

app.before_request：共享 session 的鉴权函数、请求黑白名单过滤、根据 endpoint 进行请求 j 等。蓝图场景包含 api 的请求必填字段校验，是否 json 请求校验，请求的 token 校验等。

app.errorhandler：可以用 redis 进行错误请求计数，超过一定量则进行告警。可以重定向到一个定制的错误代码页面等

app.context_processor：将一些常量按字典的格式返回，则可以在 jinja2 的模版中引用 场景：在 html 中，如果{'yao','jidan'}直接用{{jidan}}就会在页面显示 yao

app.after_request 场景：一般用于格式化响应结果，包括响应请求头，响应的格式等。

23.flask 中的四大内置对象，各有什么作用

request 请求对象,封装了客户端发送的 HTTP 请求的内容

session 用户会话,用来记住请求

g 变量 :充当者中间媒介的作用,我们可以通过它传递一些数据

current_app 代表当前的 flask 程序实例,使用时需要 flask 的程序上下文激活

24.什么是负载均衡？常见负载均衡算法

1、轮询法:将请求按顺序轮流地分配到后端服务器上，它均衡地对待后端的每一台服务器，而不关心服务器实际的连接数和当前的系统负载。

2、随机法:通过系统的随机算法，根据后端服务器的列表大小值来随机选取其中的一台服务器进行访问。由概率统计理论可以得知，随着客户端调用服务端的次数增多

3、源地址散列:根据获取客户端的 IP 地址，通过哈希函数计算得到的一个数值，用该数值对服务器列表的大小进行取模运算，得到的结果便是客服端要访问服务器的序号

4、加权轮询法:给配置高、负载低的机器配置更高的权重，让其处理更多的请；而配置低、负载高的机器，给其分配较低的权重，降低其系统负载

5 加权随机法:与加权轮询法一样，加权随机法也根据后端机器的配置，系统的负载分配不同的权重

6、最小连接数法:根据后端服务器当前的连接情况，动态地选取其中当前积压连接数最少的一台服务器来处理当前的请求，尽可能地提高后端服务的利用效率，将负责合理地分流到每一台服务器

25.Celery 是什么？常用应用场景

Celery 是一个 基于 python 开发的分布式异步消息任务队列，通过它可以轻松的实现任务的异步处理 使用场景：异步调用：那些用户不关心的但是又存在在我们 API 里面的操作 我们就可以用异步调用的方式来优化（发送邮件 或者上传头像）定时任务：定期去统计日志，数据备份，或者其他的统计任务

26.什么是 RPC，常用场景有哪些？

RPC：远程过程调用 让构建分布式计算（应用）更容易，在提供强大的远程调用能力时不损失本地调用的语义简洁性 场景：不允许较大的延迟的应用 ERP 系统回仓 2 在垂直应用中将核心和公共业务抽取出来，作为独立的服务，实现前后台逻辑分离

思考下面问题:

1. Django, Flask 和 Tornado 的区别
2. 你的后端工作职责有哪些, 如何编写项目 API 接口
3. 在公司如何与前端工程师联调
4. 后端如何反爬

第四部分 爬虫

1. 请阐述通用爬虫和聚焦爬虫的概念

   通常情况下，我们将搜索引擎使用的爬虫我们称之为通用爬虫，就像谷歌和百度的爬虫，它的作用是把整个互联网上的网页下载到本地，形成一个互联网内容的镜像备份，在对这些网页进行一定的处理（例如提取关键字、去掉广告等）之后，就可以提供一个用户检索内容的接口。一般的网站都会通过 robots 协议来告诉通用爬虫，哪些那些页面可以抓取，那些不可以抓取。

   我们做 Python 爬虫开发通常是开发聚焦爬虫，聚焦爬虫会针对某种特定的内容进行爬取，因此会对爬取的内容进行筛选和处理，而且只是抓取与特定主题需求有相关性的内容。

2. 完整的一个 url 都包含哪些内容

   一个完整的 url 包含协议、主机地址、端口、路径和资源。其中不同的协议代表不同的资源查找方式、资源传输方式；主机地址指的是存放资源的主机（服务器）的 IP 地址（域名），端口是用来区分不同的服务的，路径是指资源在主机中的具体位置。

3. 请阐述 http 协议和 https 协议的区别和联系

   HTTPS 协议需要到证书颁发机构（CA）申请证书，一般免费证书很少，需要向证书颁发机构支付费用。HTTP 协议是超文本传输协议，信息是明文传输，HTTPS 则是具有安全性的 SSL 加密传输协议。采用 HTTP 协议传输数据的客户端和服务器，客服端给服务器发送请求，服务器接收到请求会直接将数据返回。采用 HTTPS 协议的时候，客服端给服务器发送请求，服务器接收到请求后不会马上将数据返回，而是将一个证书传递给客服端（这个证书中包含了公钥），客户端信任这个证书后，客户端发出的信息会经过加密然后再发送给服务器。服务器接收到信息后会使用服务器的私钥进行解密。同样的，服务器发送给客户端的信息也会通过私钥进行加密，在客户端通过公钥进行解密。

4. robots 协议是什么

   Robots 协议，有时候也称为“爬虫协议”或“网络机器人协议”，它的全称是“网络爬虫排除标准”（Robots Exclusion Protocol），网站通过 Robots 协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取。Robots 协议对应的文本 robots.txt 通常保存在网站的根目录下，是一个使用 ASCII 码字符的纯文本文件，例如我们访问<HTTP://www.taobao.com/robots.txt>

就可以查看到淘宝的 robots 协议文件。Robots 协议并不是一个强制执行的法律法规，而是约定俗成的道德规范，所以它并不能保证网站的隐私。

目前爬虫领域仍属于拓荒阶段，因为爬取的数据就是浏览器中能够让用户看到的东西，所以“法不禁止即为许可”。但是在使用其他网站提供的数据时，必须要尊重网站的知识产权，即便是网站上的数据都由使用网站的用户提供的，但是网站平台是投入了运营成本的，当用户在注册和发布内容时，平台通常就已经获得了对数据的所有权、使用权和分发权。如果侵犯知识产权或者有损害动产的行为，在打官司时败诉的可能性是相当高的。

5. 请说出 http 请求包含哪几部分，至少 4 个常见的请求头以及含义

HTTP 请求包含请求头和请求体两个部分。请求头包含对客户端的环境描述和客户端请求信息等。请求体就是客户端要发送给服务器的数据。比如登录注册的时候，客服端需要将账号密码发送给服务器，那么这个时候的账号密码数据就可以作为请求体发送给服务器。

常见的请求头：

Host: 客户端想访问的服务器主机地址

User-Agent: 客户端的类型，客户端的软件环境

Accept: 客户端所能接收的数据类型

Accept-Language: 客户端的语言环境

Accept-Encoding: 客户端支持的数据压缩格式

6. 请说出 http 响应包含哪几部分，至少 4 个常见状态码以及含义

HTTP 响应包含响应头和响应体。响应头包含了对服务器的描述和对返回数据的描述，响应体就是服务器要给客户端返回的数据。

200：请求被成功接收

304：再次请求时，请求的内容没有发生改变

403：客户端错误，没有权限，拒绝访问

404：服务器无法根据客户端请求找到资源

500：服务器内部错误，无法完成请求

7. 用过的爬虫模块和爬虫框架都有哪些，简述优缺点

1. 下载数据 - urllib / requests / aioHTTP。

2. 解析数据 - re / lxml / beautifulsoup4（bs4）/ pyquery。

3. 缓存和持久化 - pymysql / sqlalchemy / peewee/ redis / pymongo。

4. 生成数字签名 - hashlib。

5. 序列化 - pickle / json。

6. 压缩 - zlib。

7. 调度器 - 进程（multiprocessing） / 线程（threading） / 协程（coroutine）。

8. 框架 - Scrapy 和 PySpider。

8. 讲述模拟登陆流程和爬虫中如何使用 cookie

模拟登陆可以分为两种方式，第一种使用自动化测试框架 Selenium，Selenium 调用 Chrome 浏览器来模拟人的行为，通过 Chrome 浏览器打开登陆页面，并通过获取输入账号密码的输入框，并自动的填入账号密码内容，并模拟点击登陆，即可实现登录操作。第二种使用模拟 Ajax 请求，分析页面中登录的 URL 地址，传递参数，使用这种方式模拟 API 接口进行登录。

9. 图片防盗链技术是如何实现的以及如何实现下载防盗链图片

网站防图片盗链接通常是检查获取图片的 HTTP 请求头中的 Referer 字段，该字段的作用是告诉服务器获取图片的请求是从哪个页面链接过来的，如果没有该字段或者该字段中的页面并不是本网站，那就可以认定为盗链接行为，从而通过拦截过滤器（在 Django 中是中间件）加以拒绝。知道了这一点也破解防盗链接也非常简单，就是在 HTTP 请求的请求头中添加 Referer 字段，并指定为提供图片的网站的某个有效的 URL 即可。

10. 常见的 html 文件解析方式都有哪些

正则表达式解析，速度快，如: re

XPath 解析，速度快，如:lxml

CSS 选择器解析，速度慢，如: bs4，pyquery

11. 什么是懒加载技术，如何爬取懒加载的图片

在很多网站中，如果页面中的图片内容非常多，那当用户在访问该页面时，页面中的图片不用全部都加载出来，只用加载在浏览器框中的展示的部分图片即可，因此懒加载就是当用户浏览到该图片信息时，才加载 img 中的 src 属性。实现懒加载关键点如下:1 页面中的 img 元素，如果没有 src 属性，浏览器就不会发出请求去下载图片（也就没有请求，也就提高性能），一旦通过 javascript 设置了图片路径，浏览器才会送请求。2 如何获取真正的路径，真正的路径存在元素的“data-url”属性里，当浏览器在访问到该图片时，将 data-url 地址设置到 src 属性中即可。

12.如果爬取的数据为 json 数据，如何解析

通过 Python 内置的 json 模块中的 loads 函数可以将 JSON 格式的数据解析成 Python 的字典或数组。

13.selenium+phantomjs 是什么，有什么作用

首先 Selenium 是一个自动化测试工具，利用它可以驱动浏览器执行特定的动作，如点击、表单操作等，同时还可以获取浏览器上呈现的页面源代码，做到可见即可爬。

对于一些 Javascript 动态渲染的页面来说，此种抓取方式非常有效。PhantomJS 是一个无头浏览器（headless browser），也称无界面浏览器，它是一个支持脚本编程的 Webkit 浏览器引擎。由于 PhantomJS 是无界面浏览器，爬取时不需要开界面，占用的内存较小，所以运行起来比打开界面的爬取方式更高效。除了 PhantomJS 支持无头浏览器模式，目前主流的浏览器如 Chrome 和 Firefox 也已支持 headless 浏览器模式进行无界面爬取。

14.阐述 scrapy 基本结构和工作原理

Scrapy 是一个为了爬取网站数据、提取结构性数据而编写的应用框架。

Scrapy Engine（Scrapy 引擎）：Scrapy Engine 是用来控制整个爬虫系统的数据处理流。

Scheduler（调度器）：Scheduler 维护着待爬取的 URL 队列，当调度程序从 Scrapy Engine 接受到请求时，会从待爬取的 URL 队列中取出下一个 URL 返还给他们。

Downloader（下载器）：Downloader 从 Scrapy Engine 那里得到需要下载的 URL，并向该网址发送网络请求进行页面网页，最后再将网页内容传递到 Spiders 来处理。如果需要定制更复杂的网络请求，可以通过 Downloader 中间件来实现，

Spiders（蜘蛛）：Spiders 是用户需要编辑的代码的部分。用户通过编写 spider.py 这个类实现指定要爬取的网站地址、定义网址过滤规则、解析目标数据等。 Spider 发出请求，并处理 Scrapy Engine 返回给它下载器响应数据，把解析到的数据以 item 的形式传递给 Item Pipeline，把解析到的链接传递给 Scheduler。

Item Pipeline（项目管道）：Item 定义了爬虫要抓取的数据的字段，类似于关系型数据库中表的字段名，用户编写 item.py 文件来实现这一功能。Pipeline 主要负责处理 Spider 从网页中抽取的 item，对 item 进行清洗、验证，并且将数据持久化，如将数据存入数据库或者文件。用户编写

pipeline.py 实现这一功能。

Downloader middlewares（下载器中间件）：Downloader middlewares 是位于 Scrapy Engine 和 Downloader 之间的钩子框架，主要是处理 Scrapy Engine 与 Downloader 之间的请求及响应。可以代替接收请求、处理数据的下载以及将结果响应给 Scrapy Engine。

Spider middlewares（蜘蛛中间件）：Spider middlewares 是介于 Scrapy Engine 和 Spiders 之间的钩子框架，主要是处理 Spiders 的响应输入和请求输出。可以插入自定义的代码来处理发送给 Spiders 的请求和返回 Spider 获取的响应内容和项目。

15.怎么理解进程和线程

进程是指在系统中 正在运行的一个应用程序。 进程之间是独立的，每个进程均运行在其专用且受保护的内存空间内。所有的应用程序都是通过代码来实现的（不管是什么语言），所以应用程序就可以看成是我们写好的一个代码，进程就是我们正在执行这个代码。代码中我们避免不了要声明很多的变量去保存各种各样的数据，那么这些数据就是保存在这个进程对应的内存空间中的。当进程对应的代码执行结束了，进程就结束，其对应的内存空间也会被释放。进程要想执行任务，必须得有线程（每个进程至少要有一条线程）。 一个进程（程序）的所有任务都在线程中执行。如果把进程可以看成是一个车间，线程就是车间里面的工人。车间的特点就是，每一个车间都有一个属于自己专有的空间。并且一个车间里面可以有多个工人。但是车间想要生产，那么车间中至少要一个工人。

16.如何理解同步、异步、并行、并发

同步： 多任务， 多个任务之间执行的时候要求有先后顺序，必须一个先执行完成之后，另一个才能继续执行， 只有一个主线。

异步： 多任务， 多个任务之间执行没有先后顺序，可以同时运行，执行的先后顺序不会有什么影响，存在的多条运行主线

并行和并发都可以看成是多个任务可以同时执行。但是他们之间有差别，并行才是指真正的同时执行但是要真正做到并行需要多核的支撑，并发是通过时间切片，让 cpu 快速在多个任务之间切换造成的同时执行的假象。

17.scrapy 和 scrapy-redis 有什么区别

Scrapy 是一个通用的爬虫框架，但是本身不支持分布式。Scrapy-Redis 是为了更方便地实现 Scrapy 分布式爬取，提供了以 Redis 为基础的组件。

通过 Scrapy-Redis 组件可以提供的主要功能包括：

- 爬取队列的功能
- 去重过滤的功能
- 和中断后重新爬取的功能

18.爬取过程中碰到验证码如何解决

关于这个问题，可以参考《你遇到过哪些反爬措施以及应对的手段是什么》这个问题的答案。

19.阐述 crawlspider 的定义以及用法

CrawlSpider 是 Spider 的派生类，Spider 类的设计原则是只爬取 start_url 列表中的网页，而 CrawlSpider 类定义了一些规则来提供跟进网址的方便的机制，从爬取的网页中获取网址并继续爬取的工作更适合。

CrawlSpider 的基本用法如下所示。

首先生成 scrawlspider 的命令为 ：

[scrapy genspider -t crawl 爬虫名 网站域名]

然后修改生成的 crawlspider 类，CrawlSpider 继承于蜘蛛类，除了继承过来的属性外，还提供了新的属性和方法，如：LinkExtractors 功能为提取链接，LinkExtractors 要实例化一次，并且它的 extract_links 方法会根据不同的响应多次调用提取链接；rule 属性，在规则中包含一个或多个规则对象。每个规则对爬取网站的动作定义了特定操作，如果多个规则匹配了相同的链接，则根据规则在本集合中被定义的顺序，第一个会被使用。在 rule 里需要定义 callback 方法，但由于 CrawlSpider 使用 parse 方法来实现其内部逻辑， callback 方法不能命名为 parse。

20.阐述代理是什么以及你如何使用代理

当我们需要大量的爬取网站信息时，除了切换 User-Agent 之外，另外一个重要的方式就是设置 IP 代理，以防止我们的爬虫被拒绝。设置 IP 代理通常情况下可以使用第三方的接口，也可以自己做 IP 代理池。通常情况下，一些代理网站会提供一些免费的 ip 代理，但是其稳定性和可用性很难得到保证。另外一个代理就是设置 User-Agent，用以模拟不同的浏览器的访问行为。

21.scrapy 在爬取过程中出现断网如何保证数据完整

任何程序在运行的过程中都有可能会崩溃，爬虫也不例外。当我们需要爬取的数据量很大很大的时候，爬取的过程中难免会出现各种各样的问题导致程序崩溃断掉，这个时候我们就需要记录爬虫的状态，当爬虫挂掉的时候可以恢复原来的状态继续跑。Scrapy 简单易用，效率极高，自带多线程机制。但是也正因为它的多线程机制导致在用 Scrapy 写爬虫的时候处理断点续爬很恼火。

当请求中加入了优先级属性的时候，每次 Scrapy 从请求队列中取请求的时候就会判断优先级，先取出优先级高的去访问。由于 Scrapy 默认启动十个线程。这时优先级为 100 的就会在优先级为 90 之前被取出请求队列，这样呢我们就能大体上保证爬取网页的顺序性。保证了顺序性之后呢，我们就要记录已经爬取的页数。由于发送请求、下载页面、存取数据这几个动作是顺序执行的，也就是说程序发送了这个请求不代表此时已经爬取到这一页了，只有当收到响应的时候我们才能确定我们已经获取到了数据，这时我们才能记录爬取位置。

22.要爬取的内容为动态加载，如何实现爬取

为了加速页面的加载速度，页面的很多部分都是用 JavaScript 生成的，而对于用 scrapy 爬虫来说就是一个很大的问题，因为 Scrapy 本身并没有 JavaScript 引擎，所以爬取的都是静态页面。如何分析网站上动态加载出来的内容，可以通过查看管理员工具获得网站发送的 Ajax 请求，通过调用 Ajax 请求拿到返回的 JSON 数据，返回的 JSON 数据就是网页中动态渲染的数据。另外一种方式是使用自动化测试工具 Selenium 加载 PhantomJS 或 ChromeDriver 等浏览器来加载动态内容。

23.请阐述分布式爬虫实现的原理

分布式爬虫实际上就是在多台主机上同时运行爬虫任务协同爬取，协同爬取的前提是共享同一个爬取请求队列，各台主机不需要维护自己的爬取请求队列，而是从共享队列存取请求。但每台主机有各自的调度器和下载器，所以调度和下载功能分别完成，这样就可以多台主机协同爬取，爬取效率成倍提高。

我们可以通过使用 Redis 的数据结构（如列表，集合，有序集合）来实现共享爬取请求队列

通过利用 Redis 的集合来实现分布式请求的去重

通过调度器对 Redis 的爬取请求队列进行判断，当 Redis 队列为空时，爬虫重新爬取，如果不为空，爬虫接着上次中断处继续爬取。

24.你都遇到过哪些反爬措施以及解决方式

1. 检查 HTTP 请求头。
   - Accept
   - User-Agent - 三方库 fake-useragent
   - Referer
   - Accept-Encoding
   - Accept-Language

2. 检查网站的 Cookie。
    - 有用的插件：[EditThisCookie](HTTP://www.editthiscookie.com/)
    - 如何处理脚本动态生成的 Cookie
    - 构建 Cookie 池
3. 动态生成页面内容。
    - JavaScript 逆向。
    - Selenium + PhantomJS。
4. 对访问频率进行限制。
    - 对爬虫进行限速。
    - 隐藏身份
        - IP 代理 - 快代理 / 讯代理 / 芝麻代理 / 蘑菇代理。
        - TOR - 洋葱路由，基本不考虑。
5. 填写表单或输入验证码。
    - 用 RoboBrowser 这样的工具辅助提交表单。
    - OCR（Tesseract）+ 机器学习训练，商业项目一般不考虑 。
    - 专业识别平台 - 超级鹰 / 云打码。
6. 爬虫陷阱（蜜罐）。
    - 网页上有诱使爬虫爬取的爬取的隐藏链接（陷阱或蜜罐），可以通过自动化测试工具 Selenium 调用 PhantomJS 或 ChromeDriver 来判断链接是否可见或者是否出现在可视区域来避开。

25.使用过 Scrapy 中间件吗，使用场景有哪些

Scrapy 中有以下几个中间件，分别为蜘蛛中间件（spider middleware）和下载器中间件（downloader middleware）。蜘蛛中间件是在运行过程中进行一些处理，而下载器中间件是在页面被下载时进行的一些处理。

蜘蛛中间件有以下几个函数被管理：process_spider_input 接收一个 response 对象并处理，process_spider_exception 在爬虫出现的异常时被调用，process_spider_output 当 Spider 处理 response 返回 result 时，该方法被调用，process_start_requests 当 spider 发出请求时被调用。

下载中间件有以下几个函数被管理：process_request 在请求通过下载中间件时被调用，process_response 在下载结果经过中间件时被调用，process_exception 在下载出现异常时被调用。

第五部分 数据分析和人工智能

1. 你能解释朴素贝叶斯的基本原理吗?阈值如何设定?

   朴素贝叶斯分类是一种十分简单的分类算法，叫它朴素贝叶斯分类是因为这种方法的思想真的很朴素，朴素贝叶斯的思想基础是这样的：对于给出的待分类项，求解在此项出现的条件下各个类别出现的概率，哪个最大，就认为此待分类项属于哪个类别。

2. 你能解释一下 MapReduce 是什么以及他是如何工作的吗?

   数据分片和移动计算

   MapReduce 采用计算移动到数据端方式,

   此方式极大提高数据的运算效率

   工作流程：

   第一步 Input 输入数据源

   第二步 Splitting 分割数据源，按照行数分割

   第三步 Mapping 映射 分发分割后的数据

   第四步:shuffing 其进行分组排序 此部分是透明的且不可变动

   第五步 Reducing 去重 Reducing 得到 shuffle 结果即是<K2,V2{1,1,1}>只输出其中一条,图中 Reducing 和自定义合并而非系统本身操作结果

   第六步 Final result 得到最终结果

3. 你能解释一下什么是 SVM 吗?

   支持向量机，即 supportvector machine ，它可分为线性支持向量机和非线性支持向量机。在支持向量机的分类模型中，我们首要的目的就是去找到一个最大边缘超平面（具体的下面将会介绍）。而在最大超平面的两边，有一对分别平行于最大超平面的平面，要想找到最大超平面，就得找到这样的两个平面到这个最大边缘超平面的距离最大。

4. 如何最好的选择 500 万个搜索查询的代表性样本

   评估一个样本的方法不是依据该样本的特征,而是依据其抽取的过程。对这个过程的评估涉及多个方面,比如:抽样框是否明确,样本规模有多大,样本的抽取是否严格遵循了概率抽样程序,在具体的实施过程中,有没有未应答现象,应答率是多少,等

5. 什么是 spark,以及工作流程

   Spark 是基于内存计算的大数据并行计算框架。Spark 基于内存计算，提

高了在大数据环境下数据处理的实时性，同时保证了高容错性和高可伸缩性，允许用户将 Spark 部署在大量廉价硬件之上，形成集群。

（1）任何 spark 的应用程序都包含 Driver 代码和 Executor 代码。Spark 应用程序首先在 Driver 初始化 SparkContext。

（2）SparkContext 初始化完成后，首先根据 Spark 的相关配置，向 Cluster Master 申请所需要的资源，然后在各个 Worker 结点初始化相应的 Executor。Executor 初始化完成后，Driver 将通过对 Spark 应用程序中的 RDD 代码进行解析，生成相应的 RDD graph（RDD 图），该图描述了 RDD 的相关信息及彼此之间的依赖关系。

（3）RDD 图构建完毕后，Driver 将提交给 DAGScheduler 进行解析。

（4）DAGScheduler 将划分的一系列的 Stage（TaskSet），按照 Stage 的先后顺序依次提交给底层的调度器 TaskScheduler 去执行。

（5）TaskScheduler 接收到 DAGScheduler 的 Stage 任务后，将会在集群环境中构建一个 TaskSetManager 实例来管理 Stage（TaskSet） 的生命周期。

（6）TaskSetManager 将会把相关的计算代码、数据资源文件等发送到相应的 Executor 上，并在相应的 Executor 上启动线程池执行。

（7）在 Task 执行的过程中，可能有部分应用程序涉及到 I/O 的输入输出，在每个 Executor 由相应的 BlockManager 进行管理，相关 BlockManager 的信息将会与 Driver 中的 Block tracker 进行交互和同步。

（8）在 TaskThreads 执行的过程中，如果存在运行错误、或其他影响的问题导致失败，TaskSetManager 将会默认尝试 3 次，尝试均失败后将上报 TaskScheduler，TaskScheduler 如果解决不了，再上报 DAGScheduler，DAGScheduler 将根据各个 Worker 结点的运行情况重新提交到别的 Executor 中执行。

（9）TaskThreads 执行完毕后，将把执行的结果反馈给 TaskSetManager，TaskSetManager 反馈给 TaskScheduler，TaskScheduler 再上报 DAGScheduler，DAGScheduler 将根据是否还存在待执行的 Stage，将继续循环迭代提交给 TaskScheduler 去执行。

（10）待所有的 Stage 都执行完毕后，将会最终达到应用程序的目标，或者输出到文件、或者在屏幕显示等，Driver 的本次运行过程结束，等待用户的其他指令或者关闭。

（11）在用户显式关闭 SparkContext 后，整个运行过程结束，相关的

资源被释放或回收。

6. 描述梯度提升的工作原理

以决策树为基函数的提升方法称为提升树，提升树的模型可以表示为决策树的加法模型,基函数一般是 cart 回归树，GDBT 是属于 boosting 的一员，与之对应的还有 bagging,对于 bagging 中代表算法有随机森林，boosting 中代表的有 adaboost、xgboost 等，目前数据挖掘用的比较多的还是 boosting。

7. 什么是过拟合

过拟合英文叫做 overfitting,是指为了得到一致假设而使假设变得过度严格称为过拟合.简单来说就是模型对训练数据效果很好,对预测数据效果较差,模型的泛化能力较弱.

产生过拟合的常见原因

1. 建模样本选取有误，如样本数量太少，选样方法错误，样本标签错误等，导致选取的样本数据不足以代表预定的分类规则；
2. 样本噪音干扰过大，使得机器将部分噪音认为是特征从而扰乱了预设的分类规则；
3. 假设的模型无法合理存在，或者说是假设成立的条件实际并不成立；
4. 参数太多，模型复杂度过高；
5. 对于决策树模型，如果我们对于其生长没有合理的限制，其自由生长有可能使节点只包含单纯的事件数据(event)或非事件数据(no event)，使其虽然可以完美匹配（拟合）训练数据，但是无法适应其他数据集。
6. 对于神经网络模型：a)对样本数据可能存在分类决策面不唯一，随着学习的进行,，BP 算法使权值可能收敛过于复杂的决策面；b)权值学习迭代次数足够多(Overtraining)，拟合了训练数据中的噪声和训练样例中没有代表性的特征。

过拟合的解决方法

1. 在神经网络模型中，可使用权值衰减的方法，即每次迭代过程中以某个小因子降低每个权值。
2. 选取合适的停止训练标准，使对机器的训练在合适的程度；.
3. 保留验证数据集，对训练成果进行验证；
4. 获取额外数据进行交叉验证；
5. 正则化，即在进行目标函数或代价函数优化时，在目标函数或代价函数后面加上一个正则项，一般有 L1 正则与 L2 正则等。

8. SVM 和随机森林的优缺点

SVM 的优缺点

优点:

1. 使用核函数可以向高维空间进行映射
2. 使用核函数可以解决非线性的分类
3. 分类思想很简单，就是将样本与决策面的间隔最大化
4. 分类效果较好

缺点:

1. 对大规模数据训练比较困难
2. 无法直接支持多分类，但是可以使用间接的方法来做

随机森林的优缺点

优点:

1. 简单，容易实现，计算开销小，并且它在很多现实任务中展现出来了强大的性能。它相对于 Bagging 能够收敛于更小的泛化误差，且训练效率要优于 Bagging，被誉为“代表集成学习技术水平的方法”
2. 它能够处理很高维度（feature 很多）的数据，并且不用做特征选择
3. 在训练完后，它能够给出哪些 feature 比较重要
4. 在创建随机森林的时候，对 generalization error 使用的是无偏估计，模型泛化能力强
5. 训练速度快，容易做成并行化方法
6. 在训练过程中，能够检测到 feature 间的互相影响
7. 实现比较简单
8. 对于不平衡的数据集来说，它可以平衡误差。
9. 如果有很大一部分的特征遗失，仍可以维持准确度。

缺点:

1. 随机森林已经被证明在某些噪音较大的分类或回归问题上会过拟合
2. 对于有不同取值的属性的数据，取值划分较多的属性会对随机森林产生更大的影响，所以随机森林在这种数据上产出的属性权值是不可信的。

9. 描述二值分类

所谓的二值分类又叫二元分类,其实就是二分类,即指分类结果只有两种值或两种情况,比如抛硬币只有正面和背面.很多机器学习算法初始都是用来解决二分类问题的,比如逻辑斯蒂回归,支持向量机等,在二分类问题的基础上,又可推导到多分类的情况.

二分类问题是分类问题中一个重要的课题。在二分类问题中，通常我们将两个类别分别称为正类（positive class）和负类（negative class）。其中正类通常用来表示我们所关心的那个类别，例如在异常检测中，“异常”的样本是我们所关心的，所以通常我们会把异常样本称为正样本，而正常样本称为负样本。在评估一个二分类模型的效果时，我们通常会用一个称为混淆矩阵（confusion matrix）的四格表来表示.

10.什么是交叉验证

交叉验证，英文叫 Cross Validation，是用来验证机器学习模型是否准确一种方法

基本思想很简单，就是对数据集进行分组，一部分做为训练集，另一部分做为测试集，然后，先用训练集对机器学习模型进行训练，再用测试集来测试模型，以此来对模型进行评价

交叉验证最常用对方式就是 K-CV，也就是将原始数据集分成 K 组（一般是均分），然后，将每个子集数据分别做一次验证集，得到 K 个模型，再用这 K 个模型最终的验证集的分类准确率的平均数作为此 K-CV 下分类器的性能指标。K 一般大于等于 2，实际操作时一般从 3 开始取，只有在原始数据集合数据量小的时候才会尝试取 2。

在 sklearn 中，可以通过 Model selection 里面的 cross validation 来使用次种功能

11.解释 PCA,PCA 假设,PCA 方程式

主成份分析，简写为 PCA（Principle Component Analysis）。用于提取矩阵中的最主要成分，剔除冗余数据，同时降低数据纬度。现实世界中的数据可能是多种因数叠加的结果，如果这些因数是线性叠加，PCA 就可以通过线性转化，还原这种叠加，找到最原始的数据源。

PCA 假设

1. 变量符合高斯分布（正太分布）

2. 变量之间的影响是线性的，也就是可以通过线性变化将数据还原成主要因数

3. 协方差最大的元素对应的转换向量越重要

4. 转换矩阵是正交的

PCA 的整个推导过程都是遵循上面的四条假设，如果违反了这些假设，PCA 可能作用不大，甚至有反作用，所以使用 PCA 时需要谨慎。

12.描述数据分析过程

场景分析与算法选择

数据处理

缺失值处理

数据清洗和过滤

特征工程

特征选择

特征缩放

模型训练

处理样本不均衡的问题

构件模型并训练

模型评估与优化

得分 Accuracy=(TP+TN)/(TP+TN+FN+FP)

查准率（精确率）查全率（召回率）

ROC AUC

运行效率

13.说一下逻辑斯蒂,岭回归和 lasso 回归

岭回归和 lasso 回归都是常用对线性回归模型

岭回归主要用于对有共线性的数据进行分析

实质上是一种改良的最小二乘估计法，通过放弃最小二乘法的无偏性以损失部分信息、降低精度为代价获得回归系数更为符合实际、更可靠的回归方法，对病态数据的拟合要强于最小二乘法。

lasso 回归全称是 Least absolute shrinkage and selection operator 最小绝对值收缩和选择算子。

lasso 是一种压缩估计。它通过构造一个惩罚函数得到一个较为精炼的模型，使得它压缩一些系数，同时设定一些系数为零。

14.如何判断一个新样本是否为异常值?

1.可视化方法：通过绘制坐标图的方法可以很快的发现当前的数据到底是不是离群数据。

2.数据分类法：将原先的正常数据进行二分类机器学习，目标值自己给定，计算新数据是不是异常值。

3.概率统计法：使用高斯多元求概率，和数据分类的方法类似。

15.什么是偏差-方差的抵消，怎么权衡

偏差：度量学习算法的期望预测与真实结果的偏离程度，也叫拟合能力。

方差：度量了同样大小的训练集的变动所导致的学习性能的变化，即刻画了数据扰动造成的影响。

一般来说，偏差与方差是有冲突的，这称为偏差-方差窘境。比如给定学习任务，假定我们能够控制学习算法的训练程度，则在训练不足时，学习器的拟合能力不够强，训练数据的扰动不足以使学习器产生显著变化，此时偏差主导了泛化错误率；随着训练程度的加深，学习器的拟合能力逐渐增强，训练数据发生的扰动渐渐能被学习器学到，方差逐渐主导了泛化错误率；在训练程度充分后，学习器的拟合能力已经非常强，训练数据发生的轻微扰动都能导致学习器发生显著变化，若训练数据自身的，非全局的特性被学习器学到了，则发生了过拟合。

参数或者线性的机器学习算法一般都会有一个很高的偏差和一个很低的方差。但是，非参数或者非线性的机器学习算法一般都有一个很低的偏差和一个很高的方差。所有，我们需要在这两者之间找到一个平衡点，来优化我们的算法。

比如，KNN 算法有很低的偏差和很高的方差，但是我们可以通过调整 k 的值来改变偏差和方差之间的权衡关系，从而达到一个比较平衡的状态。

因此，我们增加偏差会导致方差的减少，同理，我们增加方差会导致偏差的减少。但在实践中，我们无法去计算真正的偏差和方差值，因为我们不知道实际的目标函数。但是，作为一种方法，偏差和方差为我们提供了一种去判断机器学习算法性能的方法。

16.机器学习算法前,数据清洗的步骤

1.数据预处理：

（1）使用.replace()把一些空字符串(比如' '),特殊符号（比如 ‘?’ ）都替换成 np.nan。

（2）计算数据中特征数量数和样本数量，如果某一行或列的空值(nan)太多建议删除行或列。

（3）观察数据各个特征之间的差异是否比较大，如果是分类的话可以使用标准化进行数据的规范。

2.特征工程：

（1）如果是字典类型的数据可以转成 one-hot 编码格式(又叫独热码)。

（2）如果是文本信息，可以使用 TFIDF(重要程度)提取文本的分词重要性。

17.如何应对一个不平衡的二分类

在分类问题中如果出现某类样本过多或过少的情况我们称这是一个不平衡的分类问题。不平衡的样本类别会对模型训练产生较大的影响，为了使样本均衡化，我们往往有如下方法:

1,过采样　2,欠采样　　3,综合采样

过采样技术用来生成较少的样本，使用 SMOTE 算法可以达到上诉目的，SMOTE 全称是合成少数类过采样技术。该算法使用 knn 思想从含有较少样本类别的数据中，随机选择一个数据的 k 近邻，然后插值生成新的数据。

欠采样技术用来减少较多的样本,Tomek Links 算法可以达到上述目的，该算法会使用一些随机特性，从较多类别样本中删除一些样本，使样本达到平衡。

综合采样会综合使用上述两种方法。

18.L1 与 L2 的区别,L1 为啥具有稀疏性

L1 和 L2 范数是防止出现过拟合，降低模型复杂度的两种方法，严格上说只是 Lp 范数在 p 为 1、2 时的特殊情况。L1 即我们平时说的绝对值误差，L2 是我们平时说的平方误差。两种正则化方法都对模型的参数施加惩罚力度，在优化过程中降低一些参数的权重，从而达到降低模型复杂度的效果，但是有一些区别。

L2 范数:L2 范数使用平方误差，当某项系数小于 1 以后，惩罚力度会越来越弱，最终结果会使一些系数趋近于 0 但不会等于 0.我们常用的岭回归使用该方法

L1 范数:L1 范数使用绝对值误差，在某项系数小于 1 后仍会有一致的惩罚效果，最终会使参数项中的很多为 0。这也是为什么 L1 范数具有稀疏性的原因。

从概率模型上讲，L2 范数假设数据符合高斯分布，L1 范数假设数据符合拉普拉斯分布，从分布公式上很容易看到两种先验分布对结果产生的影响。

19.欠拟合的解决方法

欠拟合说明训练出的模型和目标数据的分布在形态上具有较大的差异，不能很好的表达当前的数据特点，具有较大的方差，同时也不具备泛化(预测)能力。

解决欠拟合从以下方面入手:

a)选择正确的模型表示。例如:对于多项式形态分布的数据选择线性模型表达，就会出现欠拟合。

b)添加更多的特征表达。当前维度的特征无法拟合出数据模型可以考虑添加更多的特征，或者将数据映射到高维空间(例如 svm 中的核函数)

c)减少正则化参数。正则化参数目的是降低模型的复杂度，防止过拟合，当出现了欠拟合应该适当降低正则化参数的影响。

20.介绍下深度学习,CNN 中的卷积和池化

CNN 既是卷积神经网络，卷积是一种数学运算，和加减乘除幂运算指数运算类似，只不过卷积运算稍微复杂一点。CNN 中的卷积操作本质上的作用是，提取数据中的特征。数据和目标值之间必然存在某种特征，是什么特征呢？假设数据和目标值之间存在一个函数，在图片识别，语音识别领域，数据很大，函数必然非常复杂，意味着系数很多，甚至上百万个系数。卷积的作用就是抽取数据之间的特征，我们计算机运算速度非常快，我们就让计算机使用卷积的运算一个个试，使用方向传播的原理告诉，计算机，正在测试的系数该调大还是调小。经过长时间运算后，计算机终于找到了合适的函数，这就是卷积操作的作用。卷积层可能会有多层，卷积核可能会很大，系数非常多，这么多系数，这么多特征，我们要关键特征，此时就用到了池化，池化操作跟卷积操作类似，不同点是，池化操作，从'池子'（相当于系数）中选择最具特征的系数（比如说最大值），池化操作相当于缩减系数。缩减系数之后，函数复杂度减小，特征更明显，便于问题的解决。

深度学习模型更复杂，体现在神经网络上，神经网络我们可以简单理解为一个个简单算法组成的。如何让这复杂的网络进行运算，找到合适的函数呢？我们使用卷积操作，说白了就是无数次的尝试（反向传播原理），一点点靠近标准答案。深度学习就是利用计算机庞大的运算能力，解方程的一个过程。方程过于复杂，我们把它叫做神经网络。

21.TensorFlow 原理,流程图,session 是啥

TensorFlow 是一个采用数据流图（data flow graphs），用于数值计算的开源软件库。数据流图用“结点”（nodes）和“线”(edges)的有向图来描述数学计算。在编写程序时，我们都是一步一步计算的，每计算完一步就可以得到一个执行结果。在 TensorFlow 中，首先需要构建一个计算图，然后按照计算图启动一个会话，在会话中完成变量赋值，计算，得到最终结果等操作。因此，可以说 TensorFlow 是一个按照计算图设计的逻辑进行计算的编程系统。

TensorFlow 的计算图可以分为两个部分：

构造部分，包含计算流图；

执行部分，通过 session 执行图中的计算。session 是模型会话，用来执行构造好的计算图，同时会话拥有和管理程序运行时的所有资源。当计算完成之后，需要通过关闭会话来帮助系统回收资源。

22.了解算法优化吗?

简单的算法优化，包括修改 sklearn 中已有算法的参数，通过 GridSearchCV 寻找最优参数

常用的优化算法包括梯度下降算法和牛顿法和拟牛顿法等

梯度下降法是最早最简单，也是最为常用的最优化方法。梯度下降法实现简单，当目标函数是凸函数时，梯度下降法的解是全局解。一般情况下，其解不保证是全局最优解，梯度下降法的速度也未必是最快的。梯度下降法的优化思想是用当前位置负梯度方向作为搜索方向，因为该方向为当前位置的最快下降方向，所以也被称为是”最速下降法“。最速下降法越接近目标值，步长越小，前进越慢。

牛顿法是一种在实数域和复数域上近似求解方程的方法。方法使用函数 f (x)的泰勒级数的前面几项来寻找方程 f (x) = 0 的根。牛顿法最大的特点就在于它的收敛速度很快。

从本质上去看，牛顿法是二阶收敛，梯度下降是一阶收敛，所以牛顿法就更快。如果更通俗地说的话，比如你想找一条最短的路径走到一个盆地的最底部，梯度下降法每次只从你当前所处位置选一个坡度最大的方向走一步，牛顿法在选择方向时，不仅会考虑坡度是否够大，还会考虑你走了一步之后，坡度是否会变得更大。所以，可以说牛顿法比梯度下降法看得更远一点，能更快地走到最底部。（牛顿法目光更加长远，所以少走弯路；相对而言，梯度下降法只考虑了局部的最优，没有全局思想。）

23.梯度下降和随机梯度下降的区别

梯度下降和随机梯度下降之间的关键区别：

1、标准梯度下降是在权值更新前对所有样例汇总误差，而随机梯度下降的权值是通过考查某个训练样例来更新的。

2、在标准梯度下降中，权值更新的每一步对多个样例求和，需要更多的计算。

3、标准梯度下降，由于使用真正的梯度，标准梯度下降对于每一次权值更新经常使用比随机梯度下降大的步长。

4、如果标准误差曲面有多个局部极小值，随机梯度下降有时可能避免陷入这些局部极小值中。

相关知识：

1、梯度下降法是一个最优化算法，通常也称为最速下降法。最速下降法是求解无约束优化问题最简单和最古老的方法之一，虽然现在已经不具有

实用性，但是许多有效算法都是以它为基础进行改进和修正而得到的。最速下降法是用负梯度方向为搜索方向的，最速下降法越接近目标值，步长越小，前进越慢。

缺点：

（1）靠近极小值时收敛速度减慢。

（2）直线搜索时可能会产生一些问题。

（3）可能会“之字形”地下降。

2、随机并行梯度下降算法，简称 SPGD 算法。作为一种无模型优化算法，比较适用于控制变量较多，受控系统比较复杂，无法建立准确数学模型的最优化控制过程。

24.从数学角度和你的个人理解完整推导和讲解 LR

LR 回归(Logistic Regression)

LR 回归，虽然这个算法从名字上来看，是回归算法，但其实际上是一个分类算法，学术界也叫它 logit regression, maximum-entropy classification (MaxEnt)或者是 the log-linear classifier。在机器学习算法中，有几十种分类器，LR 回归是其中最常用的一个。

LR 回归是在线性回归模型的基础上，使用 sigmoid 函数，将线性模型 wTx 的结果压缩到[0,1] 之间，使其拥有概率意义。 其本质仍然是一个线性模型，实现相对简单。在广告计算和推荐系统中使用频率极高，是 CTR 预估模型的基本算法。同时，LR 模型也是深度学习的基本组成单元。

LR 回归属于概率性判别式模型，之所谓是概率性模型，是因为 LR 模型是有概率意义的;之所以是判别式模型，是因为 LR 回归并没有对数据的分布进行建模，也就是说，LR 模型并不知道数据的具体分布，而是直接将判别函数，或者说是分类超平面求解了出来。

一般来说，分类算法都是求解 p(Ck|x) ，即对于一个新的样本，计算其条件概率 p(Ck|x)。这个可以看做是一个后验概率，其计算可以基于贝叶斯公式得到：p(Ck|x)=p(x|Ck)p(Ck) ,其中 p(x|Ck)是类条件概率密度，p(Ck)是类的概率先验。使用这种方法的模型，称为是生成模型，即：p(Ck|x)是由 p(x|Ck) 和 p(Ck)生成的。分类算法所得到的 p(Ck|x) 可以将输入空间划分成许多不相交的区域，这些区域之间的分隔面被称为判别函数(也称为分类面)，有了判别函数，就可以进行分类了，上面生成模型，最终也是为了得到判别函数。如果直接对判别函数进行求解，得到判别面，这种方法，就称为判别式法。LR 就属于这种方法。

25.如何对决策树进行剪枝?

决策树是充分考虑了所有的数据点而生成的复杂树，有可能出现过拟合的情况，决策树越复杂，过拟合的程度会越高。（理论来说，不应该是拟合程度越高，预测结果越准确嘛？为什么还要避免这种情况？）

考虑极端的情况，如果我们令所有的叶子节点都只含有一个数据点，那么我们能够保证所有的训练数据都能准确分类，但是很有可能得到高的预测误差，原因是将训练数据中所有的噪声数据都”准确划分”了，强化了噪声数据的作用。（形成决策树的目的作出合理的预测，尽可能有效的排除噪声数据干扰，影响正确预测的判断）

剪枝修剪分裂前后分类误差相差不大的子树，能够降低决策树的复杂度，降低过拟合出现的概率。（换句话说就是把重负累赘的子树用一个根节点进行替换，也就是说跟姐的数据意义完全可以代替又该节点衍生出的子树的所有节点的意义）

既然决定剪枝，那么怎么进行剪枝呢？两种方案：先剪枝和后剪枝

a,先剪枝说白了就是提前结束决策树的增长，跟上述决策树停止生长的方法一样。

b,后剪枝是指在决策树生长完成之后再进行剪枝的过程。

这里介绍两种后剪枝方案：

方案一：REP—错误消减剪枝

错误消减剪枝是对“基于成本复杂度的剪枝”的一种优化，但是仍然需要一个单独的测试数据集，不同的是在于这种方法可以直接使用完全诱导树对测试集中的实例进行分类，对于诱导树（什么是诱导树？）中的非叶子节树，该策略是用一个叶子节点去代替该子树，判断是否有益，如果剪枝前后，其错误率下降或者是不变，并且被剪掉的子树不包含具有相同性质的其他子树（为什么不能包含相同性质的其他子树？欢迎评论！），那么就用这个叶子节点代替这个叶子树，这个过程将一直持续进行，直至错误率出现上升的现象。简单点说：该剪枝方法是根据错误率进行剪枝，如果一棵子树修剪前后错误率没有下降，就可以认为该子树是可以修剪的。REP 剪枝需要用新的数据集，原因是如果用旧的数据集，不可能出现分裂后的错误率比分裂前错误率要高的情况。由于使用新的数据集没有参与决策树的构建，能够降低训练数据的影响，降低过拟合的程度，提高预测的准确率。

方案二：PEP-悲观剪枝法

不需要单独的数据集进行测试，而是通过训练数据集上的额错误分类数量来估算位置实力上的错误率。