一面

18’三个开放性问题

1. 有一天滴滴快车订单量少了10%，请分析原因

A、**大环境**： **季节性**（夏冬天气极端情况下，打车人比较多，天气舒适情况下，可能骑共享单车之类）、**周末**打车少（乘客：周内都上班，打车比较多，周末可能不需要赶时间或者休息不出门，打车少 车主：周期性给自己放假，不接单）、**天气**（雨雪天气出门的比较少）

B、**市场**：（**其他企业**也扩展网约车业务，分走客流量； **共享单车**的出现会减少短程订单； 也可能**共享汽车**的出现也会影响滴滴网约车的市场）

C、**付费规则的变动**：付费提升了，会减少订单量；车主提成降低，会减少接单积极性

D、**社会舆论的影响**：比如最近的新闻 “滴滴打车被偷扣99元停车费” ，负面的舆论也会影响订单量。

E、**政策影响**：如果政策对车主进一步严格要求，短期内车主会减少，车少了相应的订单量也会少。

根据不同的预测，整理数据进行验证，并实施相应解决措施，拉动订单量。

1. 请估算北京市理发店数量

**从经济学的角度来看，一个细分市场在长期竞争中如果没有出现其他较大的变量，所有企业的利润都会趋向于0，最终达到一个动态平衡。在这个假设条件下，供需是会基本平衡的。所以我从需求侧出发列出了底层公式：北京市理发店数量=市民年理发总次数/每个理发店年平均服务人数。**

下面我们该拆解这两个细分概念：市民年理发总次数和每个理发店年平均服务人数。**市民年理发总次数=男性理发人口\*男性年理发次数+女性理发人口\*女性年理发次数，每个理发店年平均服务人数=单店理发师平均数量\*单位理发师日服务人数\*365。**

再进一步拆分市民年理发总次数。记得之前在某处看到过一个数据：北京常住人口2200万人左右，而除去五分之一的无消费性理发需求者（不需要去理发店理发的人），比如秃头、长期蓄发者、和尚尼姑、父母代工的小孩子和头发生长缓慢或者子女代剪的老年人等等，可以说有1800万人口存在消费性理发需求。按照自然人口性别分布来看，再结合北京市产业特征，总体上可以认为厦门市男女性别比例较为自然状态，因此可以五五分，有消费性理发需求的男性900万人，女性900万人。按照我的日常经验，男性的理发频率要高于女性，男性一般一个月理一次发，因此一年理发12次；而女性大约三个月理一次发，一年理发4次。**因此我们可以计算出北京市市民年理发总次数=9000000\*12+9000000\*4，计算结果为144000000，也就是一亿四千四百万次的理发需求，其中男女比例为3:1。**

我们再进一步拆分每个理发店年平均服务人数。先回顾一下我们之前的公式：**每个理发店年平均服务人数=单店理发师平均数量\*单位理发师日服务人数\*365。**而根据经验，我将单店理发师平均数量定义为3人。虽然相较于我所在的较为繁华的思明区来说，3人有些少，但是其实占金字塔底层更为庞大体量的中小型理发店可能只有一两个理发师，因此平均数定义为3人可能都可能有放大之嫌。接下来是单位理发师日服务人数。**单位理发师日服务人数=日工作时间/单位顾客服务时间**，那么这里又出现了男女服务时间不同的问题，按照我的经验，男性理发服务时间一般短于女性。在这里我们将男性理发时间定义为平均30分钟，将女性理发（包含美发）时间定义为平均40分钟。之前提到了男女比例为3:1，因此按照男女比例加权之后，每个顾客所需时间平均为32.5分钟，即0.542小时。而根据我在理发时候和理发小哥的闲谈，得知他们每天上班时间在10小时左右。**用上述数据，即可计算单位理发师日服务人数=10/0.542，结果是18.45人次**，即每个理发师每天服务人数约为18.45人次，当然，不可否认的是，理发师其实也有忙闲时段之分，我们这里为了简便，按照我的常识进行了粗略的估计。**进一步可以计算出每个理发店年平均服务人数=3\*18.45\*365，结果是20202.952人次。**

准备工作基本完成，**让我们回到最开始的底层公式：北京市理发店数量=市民年理发总次数/每个理发店年平均服务人数。我们已经计算出：市民年理发总次数=144000000次，每个理发店年平均服务次数=20202.952次，所以前者除以后者，即可得到北京市理发店的数量为144000000/20202.952=7127.6，即大约为7100家！**大功告成！

其实这些问题的结果是多少并不十分重要，重要的是思路，因为**解决这类问题的能力是一个人思维缜密性和常识储备（对日常生活的关注程度）的集中体现，而这两点能力合起来，就是一个人未来在工作中分析和解决问题的潜在能力。如果思维不够缜密，是难以在面试的短时间内将一个复杂的大问题拆分为一系列小问题的；而常识储备不够多，不但很难进行数理化有逻辑的拆分，即使拆分成功，也很难对每一个数据进行较为合理的赋值，最后可能得出一个离谱的数据。**如果我计算出厦门市理发店的数量为200000家，这个数据合理吗？对一个区区500万人口的二线城市来说，答案自然是否定的。

**日常应用**

当然，这种思维方式自然不仅仅是能够应用在我们未来的面试中，其实它在生活中也有广泛的用武之地。**我们日常生活中面对的问题几乎都是复杂问题，如果我们将一个个复杂问题进行拆解，最后得到一个个能够直接进行赋值，也就是能够被直接解决的小问题，那么生活中的很多大的问题也都能迎刃而解，同时我们在解决问题的过程中也能更有针对性，避免想一出是一出，眉毛胡子一把抓。**

1. 你面前有三个箱子，其中有一个箱子有钱，其他俩里是石头，主持人打开一个箱子，发现里面是石头，这时候你得到钱的概率是多少？如果给你一次机会重新选择，你会换吗？

主持人知道空箱子的情况：换了以后概率是会提高为66.6%。原因在于最开始如果选中的是空箱子（概率是三分之二），主持人只能把剩下两个箱子中没奖的那个去掉，最后留下的那个箱子一定有奖。如果开始选了有奖的箱子（概率三分之一），那主持人随便排除剩下两个箱子中的一个，最后留下的箱子一定没奖。也就是说最开始如果选中的是空箱，那换了之后必然得奖；而最开始如果选中有奖的箱子，换了之后一定没奖。而最开始选空箱的概率是66.6%，也就是换了之后中奖的概率是66.6%。所以...换～

二面

48’主要是技术面

1. Hashmap和Hashset的区别

HashSet：

　　HashSet实现了Set接口，它不允许集合中出现重复元素。当我们提到HashSet时，第一件事就是在将对象存储在HashSet之前，要确保重写hashCode（）方法和equals（）方法，这样才能比较对象的值是否相等，确保集合中没有储存相同的对象。如果不重写上述两个方法，那么将使用下面方法默认实现：public boolean add(Object obj)方法用在Set添加元素时，如果元素值重复时返回 "false"，如果添加成功则返回"true"

HashMap：

　　HashMap实现了Map接口，Map接口对键值对进行映射。Map中不允许出现重复的键（Key）。Map接口有两个基本的实现

TreeMap和HashMap。TreeMap保存了对象的排列次序，而HashMap不能。HashMap可以有空的键值对（Key（null）-Value（null））

HashMap是非线程安全的（非Synchronize），要想实现线程安全，那么需要调用collections类的静态方法synchronizeMap（）实现。

public Object put(Object Key,Object value)方法用来将元素添加到map中。

HashSet与HashMap的区别：

|  |  |
| --- | --- |
| HashMap | HashSet |
| 实现了Map接口 | 实现Set接口 |
| 存储键值对 | 仅存储对象 |
| 调用put（）向map中添加元素 | 调用add（）方法向Set中添加元素 |
| HashMap使用键（Key）计算Hashcode | HashSet使用成员对象来计算hashcode值，  对于两个对象来说hashcode可能相同，  所以equals()方法用来判断对象的相等性，  如果两个对象不同的话，那么返回false |
| HashMap相对于HashSet较快，因为它是使用唯一的键获取对象 | HashSet较HashMap来说比较慢 |

1. MapReduce原理

**MapReduce简介**

MapReduce是一种并行可扩展计算模型，并且有较好的容错性，主要解决海量离线数据的批处理。实现下面目标

★ 易于编程

★ 良好的扩展性

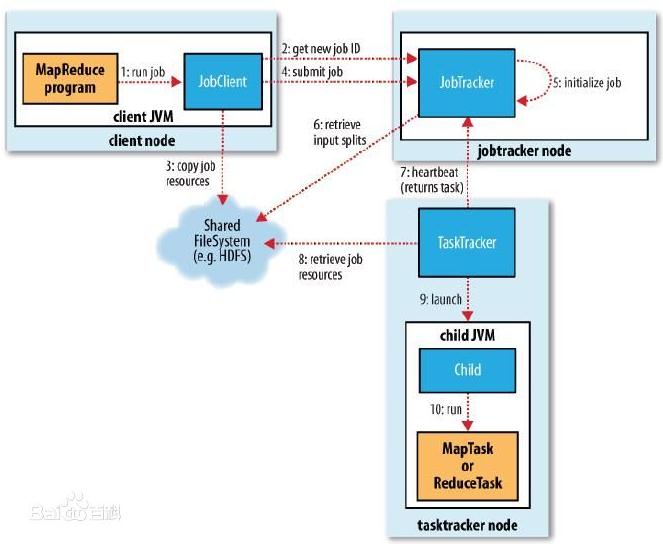
★ 高容错性

**MapReduce有哪些角色？各自的作用是什么？**

MapReduce由JobTracker和TaskTracker组成。JobTracker负责资源管理和作业控制，TaskTracker负责任务的运行。

**MapReduce程序执行流程**

程序执行流程图如下：



(1) 开发人员编写好MapReduce program，将程序打包运行。

(2) JobClient向JobTracker申请可用Job，JobTracker返回JobClient一个可用Job ID。

(3) JobClient得到Job ID后，将运行Job所需要的资源拷贝到共享文件系统HDFS中。

(4) 资源准备完备后，JobClient向JobTracker提交Job。

(5) JobTracker收到提交的Job后，初始化Job。

(6) 初始化完成后，JobTracker从HDFS中获取输入splits(作业可以该启动多少Mapper任务)。

(7) 与此同时，TaskTracker不断地向JobTracker汇报心跳信息，并且返回要执行的任务。

(8) TaskTracker得到JobTracker分配(尽量满足数据本地化)的任务后，向HDFS获取Job资源(若数据是本地的，不需拷贝数据)。

(9) 获取资源后，TaskTracker会开启JVM子进程运行任务。

注：

(3)中资源具体指什么？主要包含：

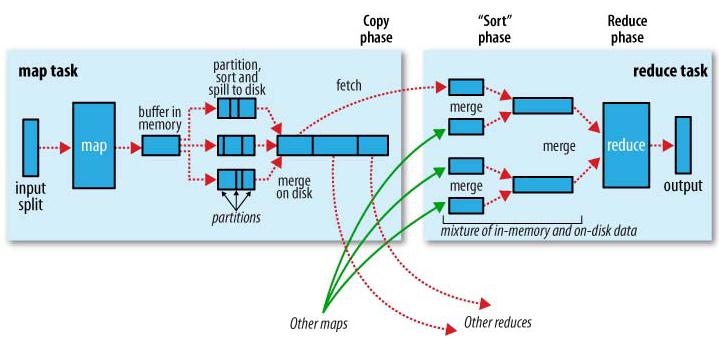
    ● 程序jar包、作业配置文件xml

    ● 输入划分信息，决定作业该启动多少个map任务

    ● 本地文件，包含依赖的第三方jar包(-libjars)、依赖的归档文件(-archives)和普通文件(-files)，如果已经上传，则不需上传

**MapReduce工作原理**

工作原理图如下：



**map task**

程序会根据InputFormat将输入文件分割成splits，每个split会作为一个map task的输入，每个map task会有一个内存缓冲区，输入数据经过map阶段处理后的中间结果会写入内存缓冲区，并且决定数据写入到哪个partitioner，当写入的数据到达内存缓冲区的的阀值(默认是0.8)，会启动一个线程将内存中的数据溢写入磁盘，同时不影响map中间结果继续写入缓冲区。在溢写过程中，MapReduce框架会对key进行排序，如果中间结果比较大，会形成多个溢写文件，最后的缓冲区数据也会全部溢写入磁盘形成一个溢写文件(最少有一个溢写文件)，如果是多个溢写文件，则最后合并所有的溢写文件为一个文件。

**reduce task**

当所有的map task完成后，每个map task会形成一个最终文件，并且该文件按区划分。reduce任务启动之前，一个map task完成后，就会启动线程来拉取map结果数据到相应的reduce task，不断地合并数据，为reduce的数据输入做准备，当所有的map task完成后，数据也拉取合并完毕后，reduce task 启动，最终将输出结果存入HDFS上。

**MapReduce中Shuffle过程**

Shuffle的过程：描述数据从map task输出到reduce task输入的这段过程。

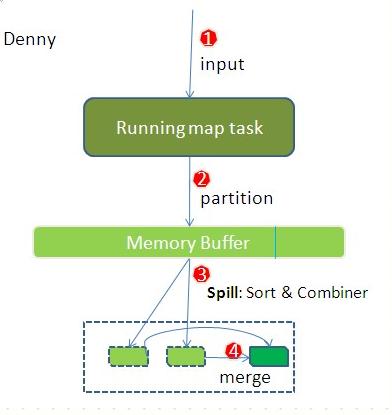
我们对Shuffle过程的期望是：

★ 完整地从map task端拉取数据到reduce task端

★ 跨界点拉取数据时，尽量减少对带宽的不必要消耗

★ 减小磁盘IO对task执行的影响

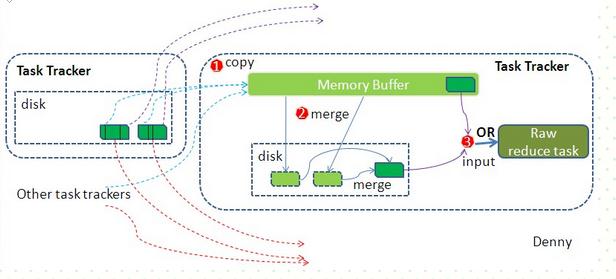
先看map端：



split被送入map task后，程序库决定数据结果数据属于哪个partitioner，写入到内存缓冲区，到达阀值，开启溢写过程，进行key排序，

如果有combiner步骤，则会对相同的key做归并处理，最终多个溢写文件合并为一个文件。

再看reduce端：



多个map task形成的最终文件的对应partitioner会被对应的reduce task拉取至内存缓冲区，对可能形成多个溢写文件合并，最终

作为resuce task的数据输入 。

**MapReduce编程主要组件**

InputFormat类：分割成多个splits和每行怎么解析。

Mapper类：对输入的每对<key,value>生成中间结果。

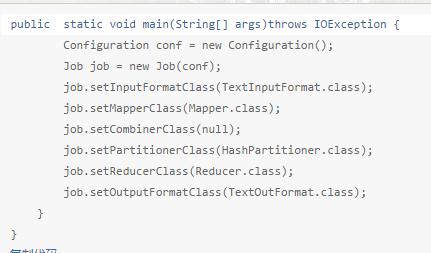
Combiner类：在map端，对相同的key进行合并。

Partitioner类：在shuffle过程中，将按照key值将中间结果分为R份，每一份都由一个reduce去完成。

Reducer类：对所有的map中间结果，进行合并。

OutputFormat类：负责输出结果格式。

编程框架如下：



**针对MapReduce的缺点，YARN解决了什么？**

MapReduce由以下缺点：

★ JobTracker挂掉，整个作业挂掉，存在单点故障

★ JobTracker既负责资源管理又负责作业控制，当作业增多时，JobTracker内存是扩展的瓶颈

★ map task全部完成后才能执行reduce task，造成资源空闲浪费

YARN设计考虑以上缺点，对MapReduce重新设计：

★ 将JobTracker职责分离，ResouceManager全局资源管理，ApplicationMaster管理作业的调度

★ 对ResouceManager做了HA设计(首先通过配置Active/Standby两个NameNode实现集群中对NameNode的热备（所谓热备就是当Active NameNode发生故障，Standby NameNode马上接手 Activy NameNode的工作）。)

★ 设计了更细粒度的抽象资源容器Container

1. 数据仓库是什么？和数据库的区别

数据仓库（Data Warehouse）是一个面向主题的（Subject Oriented）、集成的（Integrate）、相对稳定的（Non-Volatile）、反映历史变化（Time Variant）的数据集合，用于支持管理决策。  
所谓的

（1） 面向主题：指数据仓库中的数据是按照一定的主题域进行组织。

（2）集成：指对原有分散的数据库数据经过系统加工, 整理得到的消除源数据中的不一致性。

（3）相对稳定：指一旦某个数据进入数据仓库以后只需要定期的加载、刷新。

（4）反映历史变化：指通过这些信息，对企业的发展历程和未来趋势做出定量分析预测。

数据仓库建设是一个工程，是一个过程，而不是一种可以购买的产品。企业数据处理方式是以联机事务处理形式信息，并利用信息进行决策；在信息应用过程中管理信息。

数据仓库的出现，并不是要取代数据库。目前，大部分数据仓库还是用关系数据库管理系统来管理的。数据仓库与数据库的主要区别在于：

（1）数据库是面向事务的设计，数据仓库是面向主题设计的。

（2）数据库一般存储在线交易数据，数据仓库存储的一般是历史数据。

（3）数据库设计是尽量避免冗余，数据仓库在设计是有意引入冗余。

（4）数据库是为捕获数据而设计，数据仓库是为分析数据而设计。

简而言之，数据库是面向事务的设计，数据仓库是面向主题设计的。   
数据库一般存储在线交易数据，数据仓库存储的一般是历史数据。   
数据库设计是尽量避免冗余，一般采用符合范式的规则来设计，数据仓库在设计是有意引入冗余，采用反范式的方式来设计。   
数据库是为捕获数据而设计，数据仓库是为分析数据而设计，它的两个基本的元素是维表和事实表。维是看问题的角度，比如时间，部门，维表放的就是这些东西的定义，事实表里放着要查询的数据，同时有维的ID。  
单从概念上讲，有些晦涩。任何技术都是为应用服务的，结合应用可以很容易地理解。以银行业务为例。数据库是事务系统的数据平台，客户在银行做的每笔交易都会写入数据库，被记录下来，这里，可以简单地理解为用数据库记帐。数据仓库是分析系统的数据平台，它从事务系统获取数据，并做汇总、加工，为决策者提供决策的依据。比如，某银行某分行一个月发生多少交易，该分行当前存款余额是多少。如果存款又多，消费交易又多，那么该地区就有必要设立ATM了。   
显然，银行的交易量是巨大的，通常以百万甚至千万次来计算。事务系统是实时的，这就要求时效性，客户存一笔钱需要几十秒是无法忍受的，这就要求数据库只能存储很短一段时间的数据。而分析系统是事后的，它要提供关注时间段内所有的有效数据。这些数据是海量的，汇总计算起来也要慢一些，但是，只要能够提供有效的分析数据就达到目的了。   
  
数据仓库，是在数据库已经大量存在的情况下，为了进一步挖掘数据资源、为了决策需要而产生的，它决不是所谓的“大型数据库”。那么，数据仓库与传统数据库比较，有哪些不同呢?让我们先看看W.H.Inmon关于数据仓库的定义:面向主题的、集成的、与时间相关且不可修改的数据集合。  
  
“面向主题的”:传统数据库主要是为应用程序进行数据处理，未必按照同一主题存储数据;数据仓库侧重于数据分析工作，是按照主题存储的。这一点，类似于传统农贸市场与超市的区别—市场里面，白菜、萝卜、香菜会在一个摊位上，如果它们是一个小贩卖的;而超市里，白菜、萝卜、香菜则各自一块。也就是说，市场里的菜(数据)是按照小贩(应用程序)归堆(存储)的，超市里面则是按照菜的类型(同主题)归堆的。   
  
“与时间相关”:数据库保存信息的时候，并不强调一定有时间信息。数据仓库则不同，出于决策的需要，数据仓库中的数据都要标明时间属性。决策中，时间属性很重要。同样都是累计购买过九车产品的顾客，一位是最近三个月购买九车，一位是最近一年从未买过，这对于决策者意义是不同的。  
  
“不可修改”:数据仓库中的数据并不是最新的，而是来源于其它数据源。数据仓库反映的是历史信息，并不是很多数据库处理的那种日常事务数据(有的数据库例如电信计费数据库甚至处理实时信息)。因此，数据仓库中的数据是极少或根本不修改的;当然，向数据仓库添加数据是允许的。   
  
数据仓库的出现，并不是要取代数据库。目前，大部分数据仓库还是用关系数据库管理系统来管理的。可以说，数据库、数据仓库相辅相成、各有千秋。   
  
补充一下，数据仓库的方案建设的目的，是为前端查询和分析作为基础，由于有较大的冗余，所以需要的存储也较大。为了更好地为前端应用服务，数据仓库必须有如下几点优点，否则是失败的数据仓库方案。   
  
1.效率足够高。客户要求的分析数据一般分为日、周、月、季、年等，可以看出，日为周期的数据要求的效率最高，要求24小时甚至12小时内，客户能看到昨天的数据分析。由于有的企业每日的数据量很大，设计不好的数据仓库经常会出问题，延迟1-3日才能给出数据，显然不行的。   
  
2.数据质量。客户要看各种信息，肯定要准确的数据，但由于数据仓库流程至少分为3步，2次ETL，复杂的架构会更多层次，那么由于数据源有脏数据或者代码不严谨，都可以导致数据失真，客户看到错误的信息就可能导致分析出错误的决策，造成损失，而不是效益。   
  
3.扩展性。之所以有的大型数据仓库系统架构设计复杂，是因为考虑到了未来3-5年的扩展性，这样的话，客户不用太快花钱去重建数据仓库系统，就能很稳定运行。主要体现在数据建模的合理性，数据仓库方案中多出一些中间层，使海量数据流有足够的缓冲，不至于数据量大很多，就运行不起来了。

1. 什么是结构化数据

相对于结构化数据(即行数据,存储在数据库里,可以用二维表结构来逻辑表达实现的数据)而言,不方便用数据库二维逻辑表来表现的数据即称为[非结构化数据](https://www.baidu.com/s?wd=%E9%9D%9E%E7%BB%93%E6%9E%84%E5%8C%96%E6%95%B0%E6%8D%AE&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "_blank),包括所有格式的办公文档、文本、图片、XML、HTML、各类报表、图像和音频/视频信息等等。  
　　字段可根据需要扩充，即字段数目不定，可称为[半结构化数据](https://www.baidu.com/s?wd=%E5%8D%8A%E7%BB%93%E6%9E%84%E5%8C%96%E6%95%B0%E6%8D%AE&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)，例如Exchange存储的数据。  
　　非结构化数据库  
　　在信息社会，信息可以划分为两大类。一类信息能够用数据或统一的结构加以表示，我们称之为结构化数据，如数字、符号；而另一类信息无法用数字或统一的结构表示，如文本、图像、声音、网页等，我们称之为非结构化数据。结构化数据属于非结构化数据，是非结构化数据的特例  
  
　　数据清洗从名字上也看的出就是把“脏”的“洗掉”。因为数据仓库中的数据是面向某一主题的数据的集合，这些数据从多个业务系统中抽取而来而且包含历史数据，这样就避免不了有的数据是错误数据、有的数据相互之间有冲突，这些错误的或有冲突的数据显然是我们不想要的，称为“脏数据”。我们要按照一定的规则把“脏数据”“洗掉”，这就是数据清洗.而数据清洗的任务是过滤那些不符合要求的数据，将过滤的结果交给业务主管部门，确认是否过滤掉还是由业务单位修正之后再进行抽取。不符合要求的数据主要是有不完整的数据、错误的数据、重复的数据三大类。  
　　(1)不完整的数据  
　　这一类数据主要是一些应该有的信息缺失，如供应商的名称、分公司的名称、客户的区域信息缺失、业务系统中主表与明细表不能匹配等。对于这一类数据过滤出来，按缺失的内容分别写入不同Excel文件向客户提交，要求在规定的时间内补全。补全后才写入数据仓库。  
　　(2)错误的数据  
　　这一类错误产生的原因是业务系统不够健全，在接收输入后没有进行判断直接写入后台数据库造成的，比如数值数据输成全角数字字符、字符串数据后面有一个回车操作、日期格式不正确、日期越界等。这一类数据也要分类，对于类似于[全角字符](https://www.baidu.com/s?wd=%E5%85%A8%E8%A7%92%E5%AD%97%E7%AC%A6&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)、数据前后有不可见字符的问题，只能通过写[SQL语句](https://www.baidu.com/s?wd=SQL%E8%AF%AD%E5%8F%A5&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)的方式找出来，然后要求客户在业务系统修正之后抽取。日期格式不正确的或者是日期越界的这一类错误会导致ETL运行失败，这一类错误需要去业务系统数据库用SQL的方式挑出来，交给业务主管部门要求限期修正，修正之后再抽取。  
　　(3)重复的数据  
　　对于这一类数据——特别是维表中会出现这种情况——将重复数据记录的所有字段导出来，让客户确认并整理。  
　　数据清洗是一个反复的过程，不可能在几天内完成，只有不断的发现问题，解决问题。对于是否过滤，是否修正一般要求客户确认，对于过滤掉的数据，写入Excel文件或者将过滤数据写入数据表，在ETL开发的初期可以每天向业务单位发送过滤数据的邮件，促使他们尽快地修正错误,同时也可以做为将来验证数据的依据。数据清洗需要注意的是不要将有用的数据过滤掉，对于每个过滤规则认真进行验证，并要用户确认。

1. Hive SQL和SQL的区别

**1、Hive不支持等值连接**  
•SQL中对两表内联可以写成：  
•select \* from dual a, dual b where a.key = b.key;  
•Hive中应为  
•select \* from dual a join dual b on a.key = b.key;   
而不是传统的格式：  
SELECT t1.a1 as c1, t2.b1 as c2FROM t1, t2  
WHERE t1.a2 = t2.b2  
**2、分号字符**  
•分号是SQL语句结束标记，在HiveQL中也是，但是在HiveQL中，对分号的识别没有那么智慧，例如：  
•select concat(key,concat(';',key)) from dual;  
•但HiveQL在解析语句时提示：  
        FAILED: Parse Error: line 0:-1 mismatched input '<EOF>' expecting ) in function specification  
•解决的办法是，使用分号的八进制的ASCII码进行转义，那么上述语句应写成：  
•select concat(key,concat('\073',key)) from dual;  
  
**3、IS [NOT] NULL**  
•SQL中null代表空值, 值得警惕的是, 在HiveQL中String类型的字段若是空(empty)字符串, 即长度为0, 那么对它进行IS NULL的判断结果是False.  
  
**4、Hive不支持将数据插入现有的表或分区中，**  
仅支持覆盖重写整个表，示例如下：

1. INSERT OVERWRITE TABLE t1
2. SELECT \* FROM t2;

**5、hive不支持INSERT INTO 表 Values（）, UPDATE, DELETE操作**  
    这样的话，就不要很复杂的锁机制来读写数据。  
    INSERT INTO syntax is only available starting in version 0.8。INSERT INTO就是在表或分区中追加数据。  
  
**6、hive支持嵌入mapreduce程序，来处理复杂的逻辑**  
如：

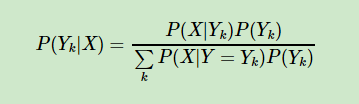
1. FROM (
2. MAP doctext USING 'python wc\_mapper.py' AS (word, cnt)
3. FROM docs
4. CLUSTER BY word
5. ) a
6. REDUCE word, cnt USING 'python wc\_reduce.py';

--doctext: 是输入  
--word, cnt: 是map程序的输出  
--CLUSTER BY: 将wordhash后，又作为reduce程序的输入  
  
并且map程序、reduce程序可以单独使用，如：

1. FROM (
2. FROM session\_table
3. SELECT sessionid, tstamp, data
4. DISTRIBUTE BY sessionid SORT BY tstamp
5. ) a
6. REDUCE sessionid, tstamp, data USING 'session\_reducer.sh';

-DISTRIBUTE BY: 用于给reduce程序分配行数据  
  
**7、hive支持将转换后的数据直接写入不同的表，还能写入分区、hdfs和本地目录**  
这样能免除多次扫描输入表的开销。

1. FROM t1
3. INSERT OVERWRITE TABLE t2
4. SELECT t3.c2, count(1)
5. FROM t3
6. WHERE t3.c1 <= 20
7. GROUP BY t3.c2
9. INSERT OVERWRITE DIRECTORY '/output\_dir'
10. SELECT t3.c2, avg(t3.c1)
11. FROM t3
12. WHERE t3.c1 > 20 AND t3.c1 <= 30
13. GROUP BY t3.c2
15. INSERT OVERWRITE LOCAL DIRECTORY '/home/dir'
16. SELECT t3.c2, sum(t3.c1)
17. FROM t3
18. WHERE t3.c1 > 30
19. GROUP BY t3.c2;
20. 朴素贝叶斯算法是什么



1. Select a, b from table … group by a会出错吗？为什么

会报错 b not in group by

1. Union all是做什么的？

union  （联合）将两个或者多个结果集合并。

在使用时，两个结果集要有相同的列，并且字段类型需要一致。

select id,name from tableA  
union all  
select name from tableB

消息 205，级别 16，状态 1，第 1 行  
使用 UNION、INTERSECT 或 EXCEPT 运算符合并的所有查询必须在其目标列表中有相同数目的表达式。

union 和union all 的区别；

union会去除结果集中重复的部分，相当于进行一个distinct（去重），并且union 会自带排序功能；

union all 会不管是否重复，都会将结果合并在一起输出，没有排序功能，只是结果集的堆叠输出。

1. Join和left join的区别

join等价于inner join内连接，是返回两个表中都有的符合条件的行。  
left join左连接，是返回左表中所有的行及右表中符合条件的行。  
right join右连接，是返回右表中所有的行及左表中符合条件的行。  
full join全连接，是返回左表中所有的行及右表中所有的行，并按条件连接。  
通常情况下，left join肯定比inner join返回的行数多。

1. 你理解一下什么是数据分析

简单来说，数据分析就是利用数据来理性思考和决策的过程。

1.网站的整体运营情况

问题：

网站的PV/UV/IP情况波动周期，是增长还是下降？

用户日活数据是怎样的，留存情况数据是怎样的？

策略：

如果整体情况偏好，细化分析，是哪些行为/内容带来的用户增长和流量？  
如果情况不好，那么哪些内容和功能是鸡肋，如何更改和优化？

2.用户的浏览习惯

问题：

用户的使用频率是怎样的？哪些时间更活跃？

用户浏览的路径是怎样的，各页面的跳转、跳出率、停留时间如何？

不同的用户对内容/功能的偏爱成都如何？

策略：

根据用户浏览行为，活跃度，可以看出网站的用户整体情况，置顶调整策略；

页面的转化、跳转从某种程度上反映了整体布局上的某些问题，很容易通过数据看出不合理的地方；

用户的使用习惯和偏好的分析则有利于实现针对性地调整，甚至个性化的推荐

3.用户的付费情况

问题：

网站的整体收益如何，影响的因素有哪些？

通过对付费数据的分析，什么样的内容更加受欢迎？

策略：

找到影响付费的关键因素，并作出决策提高销售额；

为内容的生产和付费转化提供决策。

所以，其实数据分析其实是提供决策依据的一种途径，以前我们靠经验，现在有足够的数据之后，我们可以更加直观和理性地得出一些结论。通常，一个完整的数据分析流程可以被划分为以下几个部分：

1.问题定义与细分

明确定义我们要分析的问题，找出可以量化的点，并逐步细分。比如你要分析北京的交通情况，那么细化下来你可能要着手的点有：1.几个重点路段各时间段的车辆速度分析；2.北京的各种出行方式比重；3.北京与上海、深圳等城市的拥堵情况对比；4.北京市民出行时间预测……

2.数据收集与清洗

这个很好理解，数据是才是决策的原材料，定义好问题之后，就需要找到相关的数据，并对数据的缺失值、异常值进行处理，对数据进行归一化。

3.数据建模与分析

一般我们可以做探索性的数据分析、验证性的数据分析和预测型的数据分析。探索是指我们拿到一堆数据没有头绪的时候，通过对数据进行有规律地排列展示，发现一些深层的结论；验证指的是我们本来通过经验得出一些结论，但是要通过具体的数据来进行佐证；预测则是指我们根据历史的数据，去预测未来可能产生的数据。

4.撰写数据报告

数据报告则是根据我们分析的结论，进行阐述和说明，需要利用可视化和有逻辑的数据展示来达到更好的说服力。

所以我们最终虽然是得到了一份报告，但终极目的是要去分析一件事情，得出可以支撑决策的结论。

5.执行与反馈  
就是开始干活嘛！同时干完活后需要用数据监测是否达到既定目标？如果达到了，关键因素是什么？如果没达到，问题出在哪里？

1. 你有什么优势
2. 把你做过的项目介绍一下

三面

8’技术面很短暂

1. Java中常用的数据类型说一下，一共有几个？

最近用的python

Python提供的基本数据类型主要有：布尔类型、整型、浮点型、字符串、列表、元组、集合、字典等等

1、空（None）

表示该值是一个空对象，空值是Python里一个特殊的值，用None表示。None不能理解为0，因为0是有意义的，而None是一个特殊的空值。

2、布尔类型（Boolean）

在Python中，None、任何数值类型中的0、空字符串“”、空元组()、空列表[]、空字典{}都被当作False，还有自定义类型，如果实现了nonzero()或len()方法且方法返回0或False，则其实例也被当作False，其他对象均为True   
布尔值和布尔代数的表示完全一致，一个布尔值只有True、False两种值，要么是True，要么是False，在Python中，可以直接用True、False表示布尔值（请注意大小写），也可以通过布尔运算计算出来：   
复制代码 代码如下:

布尔值还可以用and、or和not运算。 

3、整型(Int)

在Python内部对整数的处理分为普通整数和长整数，普通整数长度为机器位长，通常都是32位，超过这个范围的整数就自动当长整数处理，而长整数的范围几乎完全没限制   
Python可以处理任意大小的整数，当然包括负整数，在程序中的表示方法和数学上的写法一模一样，例如：1，100，-8080，0，等等。

4、浮点型(Float)

Python的浮点数就是数学中的小数，类似C语言中的double。   
在运算中，整数与浮点数运算的结果是浮点数   
浮点数也就是小数，之所以称为浮点数，是因为按照科学记数法表示时，一个浮点数的小数点位置是可变的，比如，1.23x109和12.3x108是相等的。浮点数可以用数学写法，如1.23，3.14，-9.01，等等。但是对于很大或很小的浮点数，就必须用科学计数法表示，把10用e替代，1.23x109就是1.23e9，或者12.3e8，0.000012可以写成1.2e-5，等等。   
整数和浮点数在计算机内部存储的方式是不同的，整数运算永远是精确的（除法难道也是精确的？是的！），而浮点数运算则可能会有四舍五入的误差。

5、字符串(String)

Python字符串即可以用单引号也可以用双引号括起来，甚至还可以用三引号括起来   
字符串是以”或”“括起来的任意文本，比如’abc’，”xyz”等等。请注意，”或”“本身只是一种表示方式，不是字符串的一部分，因此，字符串’abc’只有a，b，c这3个字符。如果’本身也是一个字符，那就可以用”“括起来，比如”I’m OK”包含的字符是I，’，m，空格，O，K这6个字符。   
如果字符串内部既包含’又包含”怎么办？可以用转义字符\来标识，比如： 代码如下:

'I\'m \"OK\"!'

表示的字符串内容是： 代码如下:

I'm "OK"!

转义字符\可以转义很多字符，比如\n表示换行，\t表示制表符，字符\本身也要转义，所以\表示的字符就是\，可以在Python的交互式命令行用print打印字符串看看： 代码如下:

>>> print 'I\'m ok.'

I'm ok.

>>> print 'I\'m learning\nPython.'

I'm learning

Python.

>>> print '\\\n\\'

\

\

如果字符串里面有很多字符都需要转义，就需要加很多\，为了简化，Python还允许用r”表示”内部的字符串默认不转义，可以自己试试：

代码如下:

>>> print '\\\t\\'

\ \

>>> print r'\\\t\\'

\\\t\\

如果字符串内部有很多换行，用\n写在一行里不好阅读，为了简化，Python允许用”’…”’的格式表示多行内容，可以自己试试：

代码如下:

>>> print '''line1

... line2

... line3'''

line1

line2

line3

上面是在交互式命令行内输入，如果写成程序，就是：   
代码如下:

print '''line1

line2

line3'''

1

2

3

多行字符串”’…”’还可以在前面加上r使用，请自行测试。

6、列表(List)

用符号[]表示列表，中间的元素可以是任何类型，用逗号分隔。list类似C语言中的数组，用于顺序存储结构   
内建函数：   
代码如下:

append(x)追加到链尾

extend(L)追加一个列表，等价于+=

insert(i,x)在位置i插入x，其余元素向后推，如果i大于列表长度，就在最后添加，如果i小于0，就在最开始添加

remove(x)删除第一个值为x的元素，如果不存在会抛出异常

reverse()反转序列

pop([i])返回并删除位置为i的元素，i默认为最后一个元素

index(x)返回x在列表中第一次出现的位置，不存在则抛出异常

count(x)返回x出现的次数

sort()排序

len(List)返回List的长度

del list[i]删除列表list中指定的第i+1个变量

切片   
切片指的是抽取序列的一部分，其形式为：list[start:end:step]。其抽取的规则是：一般默认的步长为1，但也可自定义。

7、元组(Tuple)

元组是和列表相似的数据结构，但它一旦初始化就不能更改，速度比list快，同时tuple不提供动态内存管理的功能，需理解一下规则：   
tuple可以用下标返回一个元素或子tuple   
表示只含有一个元素的tuple的方法是：(d,)后面有个逗号，用来和单独的变量相区分

8、集合(Set)

集合是无序的，不重复的元素集，类似数学中的集合，可进行逻辑运算和算术运算

9、字典(Dict)

字典是一种无序存储结构，包括关键字（key）和关键字对应的值（value）。字典的格式为：dictionary = {key:value}。关键字为不可变类型，如字符串、整数、只包含不可变对象的元组，列表等不可作为关键字。如果列表中存在关键字对，可以用dict()直接构造字典

1. Shell常用的命令说一下

1、遍历目录  
cd destination   //切换路径 destination可以是绝对路径或者相对路径（绝对路径用正斜线/作为起始，相对路径中，单点符(.)表示当前目录，双点符(..)表示当前目录的父目录）  
cd               //切换到用户主目录   CLI显示的～代表当前工作目录位于用户home目录之下  
pwd              //print work directory 打印当前工作目录  
2、文件和目录列表  
ls               //显示当前目录下的文件和目录  按字母排序（按行）可以将多个参数结3、过滤输出列表  
ls -l file       //用户可以指定特定文件的名称作为过滤器，此时只显示该文件信息  
ls -l fi?e       //ls命令能够识别标准通配符，并在过滤器中用它们进行模式匹配(文件扩展匹配)：？代表一个字符；\*代表零个或多个字符;中括号[]表示一个字符位置并给出多个选择（可列出字符也可用-给出范围）或者用!将不需要的内容排除在外；  
 4、创建文件  
touch file       //创建空文件，并将用户名作为文件的属主，此命令还可以用来改变文件的修改时间。这个操作并不需要改变文件内容。  
5、复制文件  
cp source destination       //将文件从一个位置复制到另一个位置  
cp file1 file2              //file1复制成file2,并且以file2命名  
cp -i file1 file2           //file2已存在时，加-i会提示是否覆盖已有文件  
cp -i file /home/...        //以/开头的是目录，注意不存在的目录用/开头，或者使用正确的相对路径，否则会创建成文件。  
cp -R source destination    //递归地复制整个目录的内容  
6、制表键(Tab)自动补全使用制表键自动补全的技巧在于要给shell足够的文件名信息，使其能够将需要文件同其他文件区分开  
7、链接文件链接是目录中指向真实文件位置的占位符，在linux中有两种不同类型的文件链接：  
a、符号链接：它指向存放在虚拟目录结构中某个地方的另一个文件。这两个通过符号链接在一起的文件，彼此的内容并不相同。要为一个文件创建符号链接，原始文件必须事先存在。然后利用ln命令以及-s 选项俩创建符号链接（ln -s datafile s1\_datafile），创建出来的文件是两个不同的文件，其inode编号是不同的。  
b、硬链接：会创建独立的虚拟文件，其中包含了原始文件的信息和位置。但是他们从根本上而言是同一个文件，inode号也相同。引用硬链接文件等同于引用了源文件。要创建硬链接，原始文件也必须事先存在，只不过这次使用ln命令时不需要加入额外的参数了。（ln datafile h1\_datafile）注：只能对处于同一存储媒体的文件创建硬链接。要想在不同存储媒体的文件之间创建链接，只能用符号链接。  
8、重命名文件

在linux 中，重命名文件被称为移动(moving)。mv命令可以将文件和目录移动到另一个位置或重新命名。  
mv fall fzll         //将fall文件重命名为fzll  
mv file directory    //mv命令也可以用来移动文件的位置，这一操作并不会改变文件的inode编号或时间戳  
mv file directory/file2   //mv命令可以移动位置的同时修改文件名称。  
9、删除文件

在linux中，删除(deleting)叫做移除(removing)。  
rm -i file            //会询问是否要移除文件file，输入y确认移除  
rm -i fi?e            //使用通配符删除成组文件  
rm -f file            //使用-f参数强制删除文件，小心使用  
10、创建目录  
mkdir new\_dir          //创建目录  
mkdir -p new\_dir/sub\_dir/under\_dir   //同时创建多个目录和子目录需要加入-p参数  
10、删除目录  
rmdir dir               //默认情况下,rmdir只删除空目录  
rm -ri dir              //可以在整个非空目录上使用rm命令，使用-r选项使得命令可以向下进入目录，删除其中的文件，然后删除自身。  
rm -rf dir              //此命令没有警告信息也没有信息提示，可以一口气删除目录及其所有内容，但是一个危险的工具  
11、查看文件类型  
file command.txt        //显示文件类型,eg:ASCII text、link、directory、脚本、二进制可执行文件  
12、查看整个文件  
cat file                //显示文本文件的内容

注：对于大的文件，cat命令有点繁琐。文件的文本会在显示器上一晃而过  
more file               //more命令是分页工具,会在显示每页书记后停下来,按空格显示下一页，或按回车显示下一行,输入q退出。more命令只支持文本文件中的基本移动。  
less file               //less命令是more的升级版，还支持上下键翻页。  
man less                //man 命令可以浏览对应的手册页  
13、查看部分文件  
tail file               //默认查看后10行数据  
tail -n 2 file          //输入-n修改所显示的行数 或者tail -2 file

注：-f参数是tail命令的一个突出特性。它允许你在其他进程使用该文件时查看文件内容。tail命令会保持活动状态，并不断显示添加到文件中的内容。这是实时监测系统日志的绝妙方式。  
head file               //默认查看开头10行数据  
head -n file            //修改显示的行数注：文件开头通常不会改变，因此head命令并不像tail命令那样支持-f参数特性，head命令是一种查看文件起始部分内容的便捷方法。

Linux-shell系统管理命令

1、探查进程  
ps   //查看进程 输出程序的进程ID，运行在那个终端（TTY）以及进程已用的CPU时间

2、实时监测进程  
top         //实时显示进程信息,可以在top命令运行时键入可改变top的行为。键入f允许选择对输出进行排序的字段，键入d允许修改轮询间隔,键入q可以退出top.  
3、结束进程  
linux 进程信号  
信号        名称        描述  
1           HUP         挂起    
2           INT         中断  
3           QUIT        结束运行  
9           KILL        无条件终止  
11          SEGV        段错误  
15          TERM        尽可能终止  
17          STOP        无条件停止运行但不终止  
18          TSTP        停止或暂停，但继续在后台运行  
19          CONT        在STOP或TSTP之后恢复执行  
kill pid     //终止进程ID 为PID的进程  
killall pname            //终止进程名为pname的进程,pname可用通配符  
4、监测磁盘空间  
mount         //输出当前系统上挂载的设备列表:媒体的设备文件名、媒体挂在到虚拟目录的挂载点、文件系统类型、已挂载媒体的访问状态。  
mount -t type device directory    //手动在虚拟目录中挂载设备  
eg: mount -t vfat /dev/sdb1 /run/media/centos7/    
umount [directory|device]         //卸载设备  
lsof /path/to/device/node 或lsof /path/to/mount/point //获得使用挂载设备的进程信息  
5、使用df命令  
df             //查看所有已挂载磁盘的使用情况  
6、使用du命令  
du             //显示当前目录下所有的文件、目录、和子目录的磁盘使用情况（磁盘块为单位）  
du -h          //把输出中的磁盘空间按照用户易读的形式显示，通常用M来代替兆字节，G代替吉字节  
du -c          /显示所有已列出文件总的大小  
du -s          //显示每个输出参数的总计  
7、处理数据文件  
sort file      //对文件中字符按顺序输出  
du -sh \* | sort -nr               //-r参数将结果按降序输出，管道命令将（|）将du命令的输出重定向到sort命令。  
8、搜索数据  
grep [options] pattern [file]     //grep命令在输入或指定的文件中查找包含匹配指定模式的字符的行,grep的输出就是包含了匹配模式的行。  
grep -v t file1                   //加-v参数，输出不匹配该模式的行  
grep -n t file1                   //显示匹配模式的行所在的行号,用参数-n  
grep -c t file1                   //如果只要知道有多少行含有匹配的模式，用参数-c  
grep -e t -e f file1              //可用-e参数指定每个模式，这个例子输出了含有字符t或字符f的所有行  
grep [tf] file1                   //grep用基于unix风格正则表达式匹配模式  
egrep 支持POSIX扩展正则表达式  
fgrep 支持将匹配模式指定为用换行符分隔的一列固定长度的字符串。这样就可以把这列字符串放到一个文件中，然后在fgrep命令中用其在一个大型文件中搜索字符串。  
9、压缩数据  
linux文件压缩工具  
工具         文件扩展名             描述  
bzip2           .bz2       采用Burrows-Wheeler块排序文本压缩算法和霍夫曼编码  
compress        .Z         最初的Unix文件压缩工具，已经快没人用了  
gzip            .gz        GUN压缩工具，用Lempel-Ziv编码  
zip             .zip       Windows上PK ZIP工具的Unix实现  
gzip file                    //压缩文件  
gzcat file.gz或zcat file.gz  //查看压缩过的文本文件的内容  
gunzip file.gz               //解压文件  
10、归档数据虽然zip命令能很好的将数据压缩和归档进单个文件，但它不是Unix和Linux中的标准归档工具。目前，Unix和linux上最广泛使用的归档工具是tar命令。tar命令最开始用来将文件写到磁盘设备上归档的，然而它也能把输出写到文件里。  
tar function [options] object1 object2...  
function参数定义了tar命令应该做什么  
tar -zxvf filename.tgz  //解压gzip压缩过的tar文件

1. Shell中传参用什么

"name=$1"中$1为系统提供的位置参数，$0代表程序的名称，[$1/$2/...]从1开始为传递的参数。

linux系统除了提供位置参数还提供内置参数，内置参数如下：

        $# ----传递给程序的总的参数数目

   　  $? ----上一个代码或者shell程序在shell中退出的情况，如果正常退出则返回0，反之为非0值。

         $\* ----传递给程序的所有参数组成的字符串。

        $n ----表示第几个参数，$1 表示第一个参数，$2 表示第二个参数 ... 　　$0 ----当前程序的名称

         $@----以"参数1" "参数2" ... 形式保存所有参数

         $$ ----本程序的(进程ID号)PID

         $! ----上一个命令的PID

1. Hive SQL 和 SQL 的区别
2. SQL中group by说一下

“Group By”从字面意义上理解就是根据“By”指定的规则对数据进行分组，所谓的分组就是将一个“数据集”划分成若干个“小区域”，然后针对若干个“小区域”进行数据处理。

select指定的字段必须是“分组依据字段”，其他字段若想出现在select中则必须包含在聚

select 类别, avg(数量) AS 平均值 from A group by 类别;

where 子句的作用是在对查询结果进行分组前，将不符合where条件的行去掉，即在分组之前过滤数据，where条件中不能包含聚组函数，使用where条件过滤出特定的行。

having 子句的作用是筛选满足条件的组，即在分组之后过滤数据，条件中经常包含聚组函数，使用having 条件过滤出特定的组，也可以使用多个分组标准进行分组。

1. 决策树、gbdt、xgboost

最近找工作，因为觉得以前做过比赛, 觉得自己听懂xgboost，随机森林的, 今天同实验室的一个小姐姐问了我几个问题，才发现一团乱麻，连最基本的都不懂，所以整理了一下思路。这里不重点讲定义，主要说明关系。

决策树

决策树生成包含三个步骤: 特征选择 ——> 决策树生成——> 决策树修剪:

特征选择：理解: 根据某些规则选取使决策树性能最好的特征   
1.1 信息熵   
1.2 信息增益（缺点：偏向于选取值较多的特征）   
1.3 信息增益比（基于信息增益的改进，克服了了上述信息增益的缺点）

决策树生成：   
2.1 ID3:在各个节点应用信息增益准测选择特征，递归构建决策树   
2.2 C4.5:在ID3的基础上进行了改进，其实就是使用了信息增益比准则进行特征选择， 其余不变

决策树剪枝：防止过拟合   
3.1 生成树建好后， 通过最小化loss function 去掉可以说使性能变差或无用的

因为决策树生成是一步一步的， 决策树生成是学习局部模型， 而剪枝是在整个生成树的基础上最小化loss function，所以剪枝是学习全局模型。

随机森林

随机森林是由多个决策树构成(M个）， 每个决策树单独对数据进行预测， 最终结果取这M个决策树中类别最多的那一类。不同分类器是相互独立的。   
在随机森林中，对每个决策树，加入了两个随机特征：

随机选择样本： 对于N个样本的训练集，采取有放回抽样， boostraping N个样本，这样可以保证 M个决策树的样本不同，防止over-fitting.

对特征进行采样

gbdt vs xgboost.

首先这两个都是基于boosting的。当然也具有随机森林所提到的随机性。   
关于boosting，大多数boosting 方法都是改变训练数据的概率分布，针对不同的训练数据分布调用弱分类器算法学习一系列弱分类器，一个分类器的训练是基于另一个分类器的， 所以boosting 是串行的。

不同的boosting具体规则又不同：

Adaboost:

提高那些被前一轮分类器分错的样本的权值，这样分错的样本在下一轮随机选取中被选到的概率就很大，使得分类器讲注意力放在分错的样本上

投票时，对于分类误差率较小的弱分类器赋予最大权值。

gbdt vs xgboost.

二者都是基于 上一个分类器的残差: 比如样本（x1, 10), (x2, 5), 第一分类器结果（x1, 6), (x2, 3),那么下一个分类器的训练样本就是（x1, 10-6), (x2, 5-3), 以次类推，直至残差为0，或小于某个阈值。   
不同的是：   
1. xgboost加入了第二项Ω， 模型复杂性惩罚项。   
2. 在xgboost 中特征选择是基于自己定义的评估函数，类似于信息增益。

xgboost工具支持并行。boosting不是一种串行的结构吗?怎么并行的？注意xgboost的并行不是tree粒度的并行，xgboost也是一次迭代完才能进行下一次迭代的（第t次迭代的代价函数里包含了前面t-1次迭代的预测值）。xgboost的并行是在特征粒度上的。我们知道，决策树的学习最耗时的一个步骤就是对特征的值进行排序（因为要确定最佳分割点），xgboost在训练之前，预先对数据进行了排序，然后保存为block结构，后面的迭代中重复地使用这个结构，大大减小计算量。这个block结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行。

xgboost 有用链接：[ChenTianqi xgboost论文，讲真到现在我也才看到3.2](https://arxiv.org/pdf/1603.02754.pdf)   
[雪伦大神写的详细的xgboost](http://blog.csdn.net/a819825294/article/details/51206410)   
[Weapon回答的xgboost和gbdt区别](https://www.zhihu.com/question/41354392)