整体流程：

1. 自我介绍：

* 简单介绍学校；毕业；实习的时间点
* 介绍第一份工作的主要工作内容；

宜信：

主要做一些特征工程 ；数据梳理；数据清洗

数据梳理；（咨询 ； 进件 ；审核 ；贷后（车房贷 与信贷 ））主要对以上几个系统进行梳理及表之间的相关性关系；并对字段进行可用性评估（覆盖率；准确率；时间滑动及获取难度等）

以上数据为一级数据库 不会对数据进行数据删除 防止数据信息有损失

为模型做准备：

数据清洗：异常值清洗；空置填充；离群点检测；歧义值清洗

异常值清洗（年龄过大 或者过小 ；性别中有汉字 ；学历婚姻状况中有根据时间的推进含义不一致情况）等 ；空置清洗：根据均值（或者 -999） 填充 ；

后面主要做了一些风控模型（申请风险模型 和收益风险模型）还做过一些小模型（基于规则的贷后评分 和高精端客户转化模型）

* 介绍第二份工作 搜狗

识别pc网页库中的无线版式页面，在搜索时屏蔽。

了解html 和css 在无线 pc 页面展现上面的差异，主要通过 url ； html ；渲染；content ；版式 几个角度构建特征集合 应用树模型进行特征选择。构建准确率（100%）的响应性规则

1. 介绍在两个公司印象笔记深刻的项目：

宜信：申请风险模型 和收益风险模型

申请风险评分模型：

* 数据范围：主要根据城市信贷为基础：精英贷、新薪贷、新薪宜楼贷、精英贷（银行合作）、新薪贷（银行合作）
* 时间范围：20141001～20160331   由于之后的跟跑选择16年之后的数据 ，要大约有一年左右的表现期。
* 标签：

主要根据系统里面的是否放款：好人与拒贷（拒贷中的拒贷码进行好坏人的定义）

拒贷 并不是全部定义为坏人 ：主要根据拒贷原因进行统计，拒贷原因中主要是逾期相关及无还款能力低等作为原则上的坏人，有一些欺诈用户不在标签其中

因为 欺诈用户的信息模型是学习不出来的。

|  |  |
| --- | --- |
| 坏人 | 163728.0 |
| 不确定为坏人 | 484158.0 |
| 好人 | 647234.0 |

* 特征选择：

首先根据覆盖率 distinct值进行剔除覆盖率较低且

特征平稳——训练集和测试集的好坏趋势一致：主要画出 训练集测试集的密度图：查看相关差异性

有区分性——好坏趋势有规律，有差异 主要根据WOE进行划分

模型选择：要求 有可解释性 虽然用GBDT xgboost 跑过，但是最后展示出来的结果要用LR 解释性强

针对LR 做的特征工程：

连续址：

离散化、分箱  WOE 离散化

woe反映的是在自变量每个分组下违约用户对正常用户占比和总体中违约用户对正常用户占比之间的差异；从而可以直观的认为woe蕴含了自变量取值对于目标变量（违约概率）的影响

主要根据好人比例图进行划分bin值：横坐标 为此特征排序的占比，纵坐标为好人占比，

在分bin时候，主要保证每个小bin数据量不小于 2000  通话好人比例图，可以找到连续值的好人占比较高的特征区间即可以找到分割点

离散值：one hot encoding

运用LR 模型：L1 正则 主要调节lamda

收益模型：标签：有净收益最小期数问题

总收入(all\_in)=sum（0-i）（每期实还本金+每期实还利息+每期服务费）（求第 i期 之前所有实还总和   每一期都会发生变化）

总支出(all\_out)=本金+资金成本-前期服务费     (固定)

净收益(net\_revenue)=总收入-总支出。

结果评价：

P R F1

Roc ks FindbestThreshold

资本市场建模：

客群：高净值客户  潜客池  全部数据

时间：15年  ：17年 4月底    购买资本市场

标签：未来30天 是否 会购买  资本市场

交易：

产品类型

币种

产品名称

预约：

|  |
| --- |
| 预约状态  预约来源  陪访：  陪访评价  沟通效果  数据来源  曾经参与过的投 |

识别pc网页库中的无线版式页面，在搜索时屏蔽：

特征构建：

主要从三个方面进行特征抽取

URL ； HTML ；渲染差异

主要了解HTML PC端和无线端差异：

主要是html 的head标签； body的a标签 ；页面板式（渲染之后的元素宽度高度等信息）；通栏信息

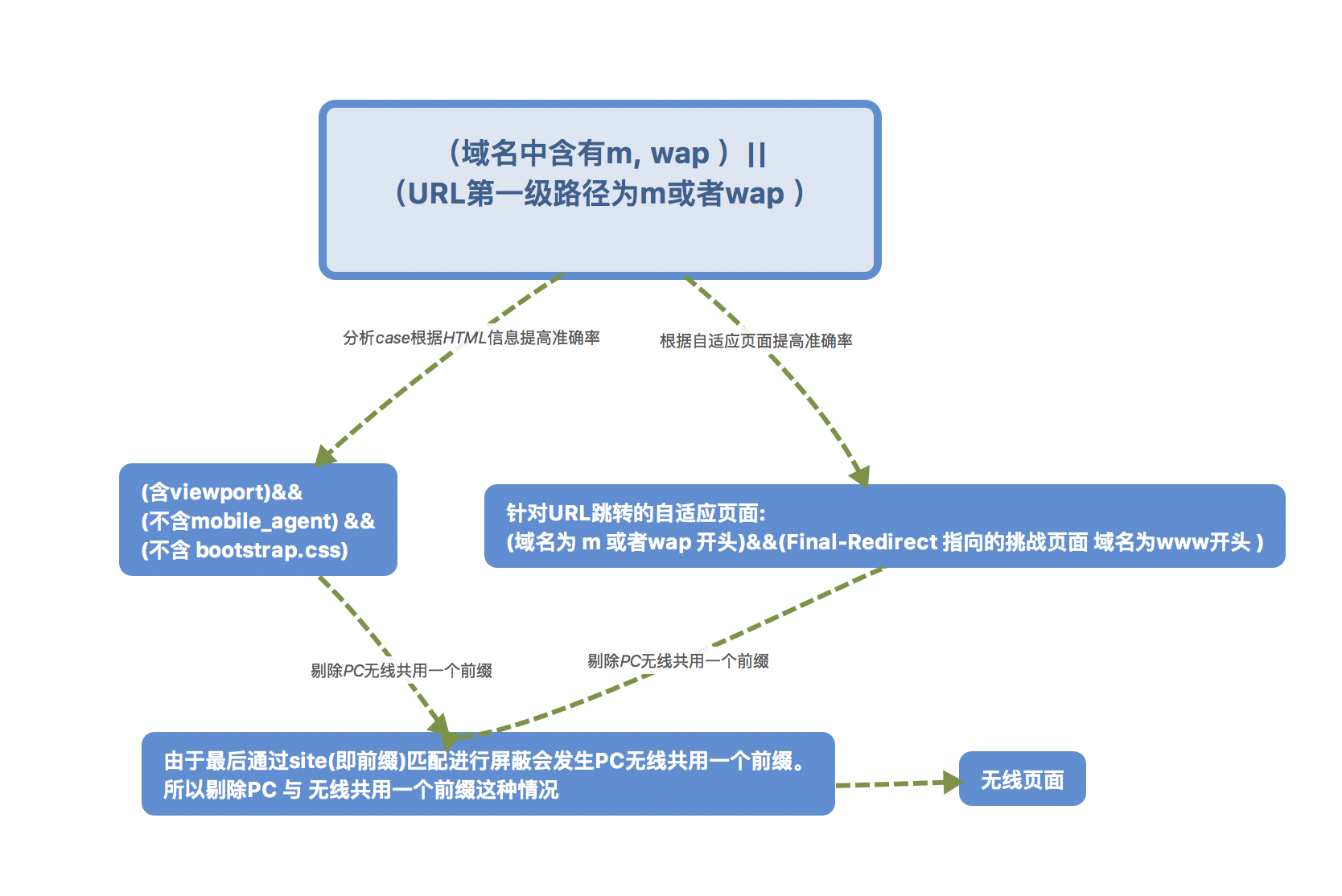
|  |  |
| --- | --- |
| title\_feature | Title 中含有3g版；手机版；触屏版；行動版 等 |
| maxWidth | 最大宽度 |
| getbannercnt | 通栏个数 |
| getPercent | 通栏占比 |
| isBanner | 是否是通栏页面 |
| viewportBaseCount | 包含viewport 的个数 |
| apple\_touch | 包含apple-touch-fullscreen |
| apple\_mobile | 在iOS中有两个meta值，apple-mobile-web-app-capable和apple-mobile-web-app-status-bar-style，这两个会让网页内容以应用程序风格显示，并使状态栏透明。包含 apple-mobile-web-app 标签. value=‘yes’ |
| mobileagentcount | 包含 mobile-agent的个数 |
| userscalablevalue | 放缩大小 |
| handheldfriendly | handheldfriendly. value=true |
| mobileoptimized | MobileOptimized |
| href\_skip | 1 ！only pc and notstartpc 2: only\_wap 4:：都不为null ，only start with pc  3:：都不为null ，only start with wap 5:：都不为null ，都匹配 wap长度长 other 6 |
| isadaption | .\*(width\\s\*=\\s\*\\d{3,4}).\* |
| maximum\_scale |  |
| URL\_has\_mwap | 域名中包含 m，wap 标签 |
| URL\_has\_3g,4g,mobi | 域名中包含 3g 4g mobil 关键字 |
| First\_path\_mwao | 第一级目录中包含map |
| Has\_**viewport** | **含<meta name="****viewport" value="...">标签的页** |
| Has\_ apple-mobile-web-app | 含 name="apple-mobile-web-app" value="...">标签 |
| Not\_has\_ mobile-agent | 不含<meta http-equiv="mobile-agent" value="... |
|  |  |

构建40维度左右的特征集合

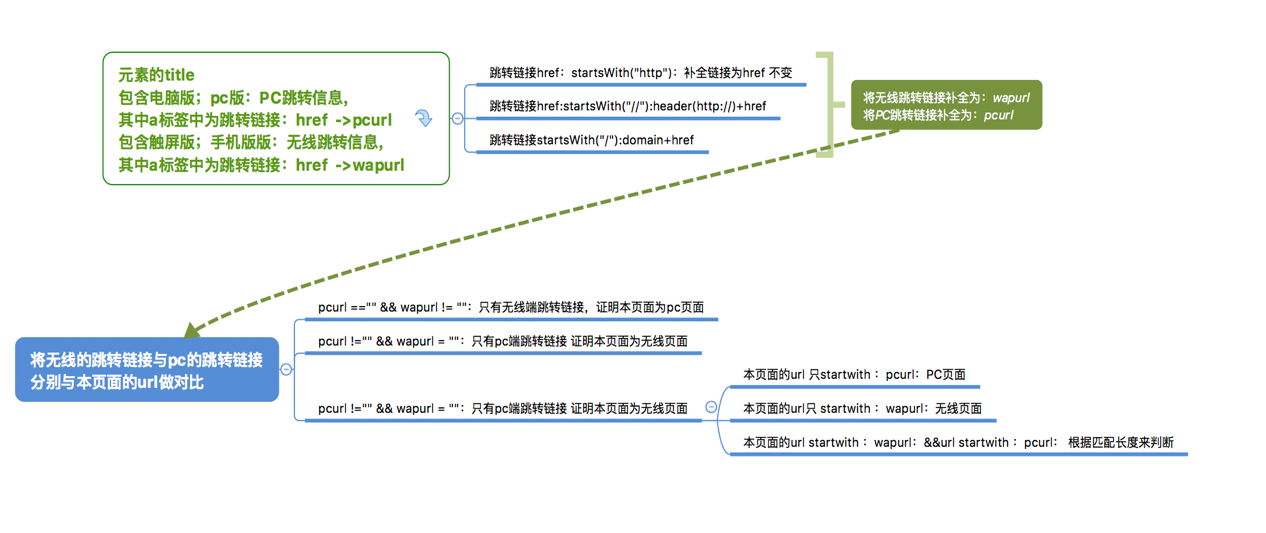
构建出一批纯度较高的wap pc url 各抽取2万数据集合 将数据根据以上model 跑出base line 准确率 88% 但是上线要求 ：准确率 100% 由于做的是基础数据库的删选，所有不能黑名单 不能加载 所以根据 gbdt 二叉树结构进行特征构建。

主要构建以下规则：

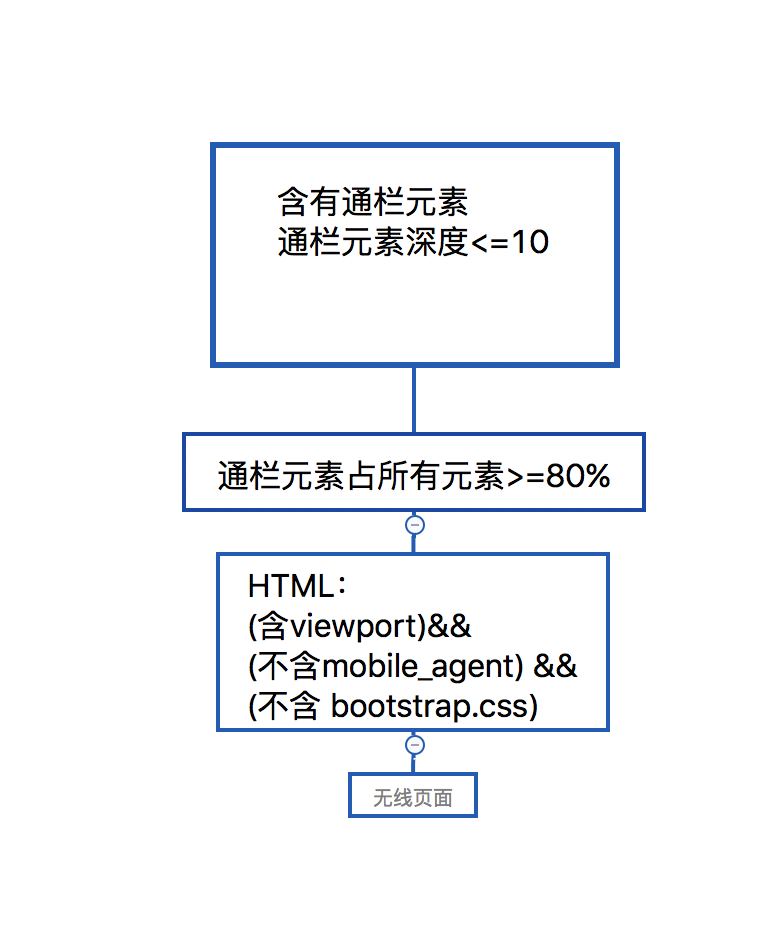
1:



2:



3:



分析case 优化规则

sbs 实验：

|  |  |
| --- | --- |
| 标注数据来源 | baidu、sogou wap日志7天数据（2018.03.06 – 2018.03.12） |
| 抽样方法 | 随机抽取 |
| 数据量 | 共60000条数据（其中baidu 30000、sogou 30000）diff查询967条，影响面1.61%，百度覆盖面：443/30000 =1.47%， 搜狗覆盖面：524/30000 =1.74% |



1. 算法推导：

* LR：本身是回归问题；加入映射来转化为分类问题 将线性不可分的问题转化为线性可分问题
* 这个映射为sigmoid 函数：sigmoid 的由来（贝叶斯 ；伯努利 角度可以推导 ok）
* 预测函数已有；此时求解参数 目的 最小化经验风险 防止过拟合 加入正则，转化为结构风险，（最大对数似然 求导 ok 求得迭代结构 sum（h（xℹ） -yi）／M ok）
* L1 L2 正则 降低模型复杂度 原理：ok

<https://liam0205.me/2017/03/30/L1-and-L2-regularizer/>

L1正则化不可导，怎么求解？坐标轴下降法（按照每个坐标轴一个个使其收敛）

* 牛顿法 梯度下降法 坐标轴下降法（半ok 状态）<https://blog.csdn.net/u014688145/article/details/53688585>
* ROC KS +F1 理论及含义 ok
* Xgboost +gbdt：差异 + 推导 基本ok

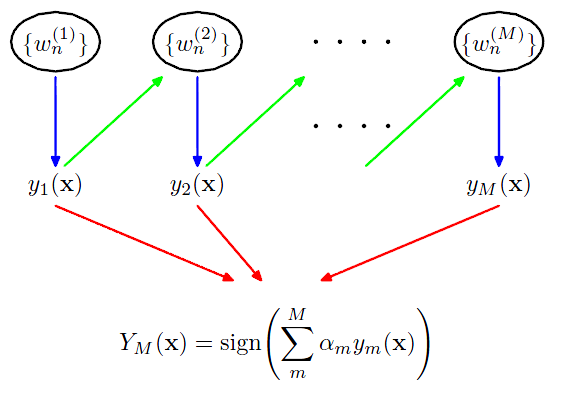
gbdt:

首先 xgboost 是加法模型：

<https://blog.csdn.net/zqxnum1/article/details/44337557>

<https://www.jianshu.com/p/d2109fcd8a2e>

https://www.cnblogs.com/LeftNotEasy/archive/2011/01/02/machine-learning-boosting-and-gradient-boosting.html



基本都是弱分类器的加和 改变下一个模型输入数据的权重，将每个模型进行求和得到最终的 sign（sum（））

每个模型的 参数：由于目的是想每个model 的损失函数下降的速度的最大，所以每次损失函数的优化方向应该为上次损失函数的梯度方向 此时顾为最大损失函数优化方向：

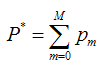
假设我们的模型能够用下面的函数来表示，P表示参数，可能有多个参数组成，P = {p0,p1,p2….}，F(x;P)表示以P为参数的x的函数，也就是我们的预测函数。我们的模型是由多个模型加起来的，β表示每个模型的权重，α表示模型里面的参数。为了优化F，我们就可以优化{β,α}也就是P。

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022146375783.png)

    我们还是用P来表示模型的参数，可以得到，Φ(P)表示P的likelihood函数，也就是模型F(x;P)的loss函数，Φ(P)=…后面的一块看起来很复杂，只要理解成是一个损失函数就行了，不要被吓跑了。

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022146384222.png)

   既然模型(F(x;P))是可加的，对于参数P，我们也可以得到下面的式子：

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022146459974.png)

   这样优化P的过程，就可以是一个梯度下降的过程了，假设当前已经得到了m-1个模型，想要得到第m个模型的时候，我们首先对前m-1个模型求梯度。得到最快下降的方向，gm就是最快下降的方向。

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022146545603.png)

    这里有一个很重要的假设，**对于求出的前m-1个模型，我们认为是已知的了，不要去改变它，而我们的目标是放在之后的模型建立上。**就像做事情的时候，之前做错的事就没有后悔药吃了，只有努力在之后的事情上别犯错：

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022146553344.png)

    我们得到的新的模型就是，它就在P似然函数的梯度方向。ρ是在梯度方向上下降的距离。

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022146556343.png)

    我们最终可以通过优化下面的式子来得到最优的ρ：

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022146592566.png)

**通用的Gradient Descent Boosting的框架：**

   下面我将推导一下Gradient Descent方法的通用形式，之前讨论过的：

[mage](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022147098445.png)

对于模型的参数{β,α}，我们可以用下面的式子来进行表示，这个式子的意思是，对于N个样本点(xi,yi)计算其在模型F(x;α,β)下的损失函数，最优的{α,β}就是能够使得这个损失函数最小的{α,β}。

[mage](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022147102282.png) 表示两个m维的参数：

[mage](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/20110102214721778.png)

写成梯度下降的方式就是下面的形式，也就是我们将要得到的模型fm(x)的参数{αm,βm}能够使得fm的方向是之前得到的模型Fm-1(x)的损失函数下降最快的方向：

[mage](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022147252334.png)

   对于每一个数据点xi都可以得到一个gm(xi)，最终我们可以得到一个完整梯度下降方向

[mage](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022147267251.png)

[mage](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022147489367.png)

为了使得fm(x)能够在gm(x)的方向上，我们可以优化下面的式子得到，可以使用最小二乘法：

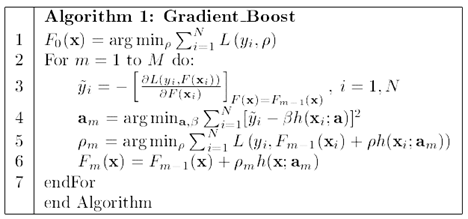
[mage](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022147491743.png)

得到了α的基础上，然后可以得到βm。  [mage](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022147533332.png)

最终合并到模型中：

[mage](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022147578966.png)

    算法的流程图如下

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/201101/201101022148004765.png)

xgboost：

https://www.zhihu.com/question/41354392

在XGBoost的实现， 为了便于求解，对loss function基于Taylor Expansion进行了变换：   
  
在变换完之后的形式里， 就是为了优化loss function，待更新优化的变量（这里的变量是一个广义的描述）。  
上面的loss function是针对一个样本而言的，所以，对于样本全集来说，loss function的形式是：   
  
对这个loss function进行优化的过程，实际上就是对第k个树结构进行分裂，找到启发式的最优树结构的过程。而每次分裂，对应于将属于一个叶结点（初始情况下只有一个叶结点，即根结点）下的训练样本分配到分裂出的两个新叶结点上，每个叶结点上的训练样本都会对应一个模型学出的概率值，而loss function本身满足样本之间的累加特性，所以，可以通过将分裂前的叶结点上样本的loss function和与分裂之后的两个新叶结点上的样本的loss function之和进行对比，从而找到可用的分裂特征以及特征分裂点。  
而每个叶结点上都会附著一个weight，这个weight会用于对落在这个叶结点上的样本打分使用，所以叶结点weight的赋值，也会影响到loss function的变化。基于这种考虑，也许将loss function从样本维度转移到叶结点维度也许更为自然，于是就有了下面的形式：   
  
上面的loss function，本质上是一个包含T(T对应于Tree当前的叶子结点的个数)个自变量的二次函数，这也是一个convex function，所以，可以通过求函数极值点的方式获得最优解析解（偏导数为0的点对应于极值点），其形如下：   
  
现在，我们可以把求解过程串接梳理一下：

I. 对loss function进行二阶Taylor Expansion，展开以后的形式里，当前待学习的Tree是变量，需要进行优化求解。   
II. Tree的优化过程，包括两个环节：   
I). 枚举每个叶结点上的特征潜在的分裂点   
II). 对每个潜在的分裂点，计算如果以这个分裂点对叶结点进行分割以后，分割前和分割后的loss function的变化情况。  
因为Loss Function满足累积性(对MLE取log的好处)，并且每个叶结点对应的weight的求取是独立于其他叶结点的（只跟落在这个叶结点上的样本有关），所以，不同叶结点上的loss function满足单调累加性，只要保证每个叶结点上的样本累积loss function最小化，整体样本集的loss function也就最小化了。  
而给定一个叶结点，可以通过求取解析解计算出这个叶结点上样本集的loss function最小值。

有了上面的两个环节，就可以找出基于当前树结构，最优的分裂点，完成Tree结构的优化。   
这就是完整的求解思路。有了这个求解思路的介绍，我们就可以切入到具体实现细节了。   
注意，实际的求解过程中，为了避免过拟合，会在Loss Function加入对叶结点weight以及叶结点个数的正则项，所以具体的优化细节会有微调，不过这已经不再影响问题的本质，所以此处不再展开介绍。

1. 数据结构 算法推导

快排

堆排 ；桶排序

编辑距离

随机采样

手写算法：

1. 机动车摇号：不用考虑海量数据的情况，有一个用户的池子，存放数据( 用户id \t 用户开始摇号的时间 )，设计一个随机抽取的程序，使用户被抽中的概率与开始摇号距离现在的年数成正比，如普通被抽中的概率为p，一个摇了两年号的用户被抽中的概率为2p

<http://www.gocalf.com/blog/weighted-random-selection.html>

2、给定一个有序的数组，对该数组进行一次rotation，如有序数组[1,2,3,4,5]，从4的位置rotation得到数据[4,5,1,2,3]，设计算法对rotation后的数组返回指定数的下标，如1，则返回下标2 ok

3、给定一个整型数组，含有正负数，求取子数组的最大和，返回最大和 ok

4、单链表删除指定节点 ok

5、单链表逆序 ok

https://blog.csdn.net/autumn20080101/article/details/7607148

6、不考虑海量数据的情况，给定数据文件(两列数groupid \t label)，label非0即1，数据记录的顺序已固定，相同groupid的数据在一起，定义逆序数是0在1前面的个数，正序数是1在0前面的个数，设计算法求取所有group下的正序数之和/逆序数之和的比

行

7、快排实现 ok

8、最小编辑距离 ok

递归 +非递归 python 版本

https://blog.csdn.net/luo123n/article/details/9999481

9、构建搜索树

https://www.cnblogs.com/coffeeSS/p/5452719.html

模型：

LSTM与RNN的区别

LSTM的原理、公式

lightgbm的优化

1:LR 牛顿法 和梯度下降 与随机剃度下降区别，凸函数如何求

针对一个点和全部点之间的差别 一般用是一个batch 的数据求解剃度方向；

牛顿 是由于求解还森矩阵 难求

二阶导数 ；如何没有 则做矩阵 正定来判定

2:卡方验证 选择特征的逻辑 交叉熵

3: GBDT 随机剃度的原因：为何要用随机剃度

GBDT是以决策树为基学习器的迭代算法，注意GBDT里的决策树都是回归树而不是分类树。Boost是”提升”的意思，一般Boosting算法都是一个迭代的过程，每一次新的训练都是为了改进上一次的结果。   
GBDT的核心就在于：每一棵树学的是之前所有树结论和的残差，这个残差就是一个加预测值后能得真实值的累加量。比如A的真实年龄是18岁，但第一棵树的预测年龄是12岁，差了6岁，即残差为6岁。那么在第二棵树里我们把A的年龄设为6岁去学习，如果第二棵树真的能把A分到6岁的叶子节点，那累加两棵树的结论就是A的真实年龄；如果第二棵树的结论是5岁，则A仍然存在1岁的残差，第三棵树里A的年龄就变成1岁，继续学习。   
GBDT优点是适用面广，离散或连续的数据都可以处理，几乎可用于所有回归问题（线性/非线性），亦可用于二分类问题（设定阈值，大于阈值为正例，反之为负例）。缺点是由于弱分类器的串行依赖，导致难以并行训练数据。

4:

蓄水池算法：

**ArrayList**<String> urls\_arr = **new** ArrayList<>();

**public** **void** **reduce**(**Text** key, Iterable<Text> values, **Context** context) **throws** **IOException**, **InterruptedException** {

row++;

**Random** **random** = **new** Random();

**String** **url** = key.toString();

**if** (urls\_arr.size() < sample\_num) {

urls\_arr.add(url);

} **else** {

**int** **index** = random.nextInt(row);

**if** (index < sample\_num) {

urls\_arr.set(index, url);

}

}

}