比值假设检验

# 基础知识部分

## 数据库部分

### 数据倾斜解决方案：

通常出现在groupby 和 join的时候，数据分布不均匀，某一个值的数据量较大，在shuffle阶段被发往同一个reduce，导致该reduce速度较慢，整条任务都在等这个一个reduce完成，内存资源无法释放出去

处理方式：

1、**手动分割大查询**

对于可能产生数据倾斜的大查询，手动分割查询可以是一个有效的方法：

2、分阶段执行：先对数据进行预聚合或预处理，减少数据倾斜的影响。

3、使用临时表：将数据先分布到临时表中，再从临时表查询。

#### 增加系统开销

I/O成本：打散操作可能增加数据流动的I/O成本，因为原本可能局部处理的数据现在需要在多个节点之间传输。

存储成本：在某些情况下，为了实现有效的打散，可能需要额外存储中间结果，这将增加存储成本。

### Sql优化

sql优化看运⾏环境，可以分为mysql和Hive，前者是数据库查询优化，后者基于MapReduce。互联⽹分析师多是基于Hive查询数据，故下⽂针对Hive如何优化进⾏分析。

选择合适的数据粒度，是sql优化的基础。例如选择⽤户粒度的Hive表，比起访问pv粒度的Hive表，数据量要⼩很多，sql查询也更快。

基本的查询优化

1、根据需求选择合适的数据粒度

2、避免使用 SELECT \*: 只获取必要的列，避免不必要的数据读取。

3、使用WHERE过滤，合理使用子查询，尽量将结果进行过滤，减少数据量。

4、语法优化在处理大量数据和多列时，**GROUP BY** 可能比 **DISTINCT** 更优，特别是当只需要去重少数几个字段而不是所有字段时，使用GROUP BY与ORDER BY时考虑是否真正需要，exists替代in

5、数据倾斜的处理

合理设置分区表，提高查询速度

利用桶表优化

拆分进行子查询

### Join方式

Inner join 内连接

full outer join 全外连接，全连接 返回两表所有的列

left join

---下面这俩只返回左表

left semi join

left anti join

Left Anti Join，也称作 Left Anti-Semi Join，是 Left Semi Join 的逻辑补充。它返回左表中没有在右表中找到匹配的行的所有行。

----以下的两个连接不用指定on

cross join 就是把两个表所有的行进行一个组合 m 行笛卡尔积 n行 则就是m\*n 行的结果

natural join 自然连接 自动识别表中的相同字段，并且自动用这些相同字段的值进行内连接

### 抽样取N个值

随机排序

order by rand() limit n

抽样取

文本

描述已自动生成

### 取上月

图形用户界面, 文本, 应用程序

描述已自动生成

### 手撕分位数

从小到大排列 ，然后从左往右第25%

25%分位数通常指的是下25%分位数（即第25百分位数）。这是指数据集中有25%的数据小于或等于这个值，剩下的75%的数据大于这个值。

但是

**上25%分位数（即第75百分位数）和下25%分位数（即第25百分位数）是不一样的。**

\*\*上25%分位数（第75百分位数）\*\*表示的是数据集里有75%的数据小于或等于这个值，而有25%的数据大于这个值。

\*\*下25%分位数（第25百分位数）\*\*表示的是数据集里有25%的数据小于或等于这个值，而有75%的数据大于这个值。

换句话说，上25%分位数是比较高的值，而下25%分位数是比较低的值。

with tmp1 as (

select

money\_sum,row\_number() over(partition by 1 order by money\_sum) as rn

,(0.5\*count() over()+1) as mid\_num

from

bip\_member.hmy\_storeidx\_source

)

select

money\_sum\_low+(money\_sum\_high-money\_sum\_low)\*(mid\_num-floor(mid\_num))

from

(

select

money\_sum as money\_sum\_low,mid\_num

from tmp1

where rn=floor(mid\_num)

)

cross join

(

select

money\_sum as money\_sum\_high

from tmp1

where rn=ceil(mid\_num)

)

;

### 连续七天登录语法：

各种连续的都是要做辅助列 row\_number 然后用原本的时间列去减 再group by

### 转置（通常通过列转行+unionall实现）

SELECT

max(case when number=1 then number end) AS is\_number1

,max(case when number=2 then number end) AS is\_number2

,max(case when number=3 then number end) AS is\_number3

,max(case when number=4 then number end) AS is\_number4

,max(case when number=5 then number end) AS is\_number5

FROM table\_number

### 默认窗口范围

OVER (PARTITION BY ) 如果没有order by字句 和range between 字句的话 range between默认范围是什么

在SQL的窗口函数中，OVER子句可以与PARTITION BY, ORDER BY, 和RANGE/ROWS/GROUPS between子句一起使用来定义窗口的行为和边界。如果在使用OVER子句时省略了ORDER BY和RANGE/ROWS/GROUPS between子句，窗口的默认行为会根据所使用的函数和具体的SQL数据库的实现有所不同。以下是一些通用的规则和概念：

无ORDER BY子句：

如果OVER子句中没有ORDER BY子句，窗口函数将考虑窗口中的所有行为相等，这意味着每个窗口是一个平等的集合，其中没有行的顺序。在这种情况下，RANGE BETWEEN、ROWS BETWEEN或GROUPS BETWEEN子句（如果有的话）的默认行为通常如下：

RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

这意味着窗口包含从分区的第一行到最后一行的所有行，无论RANGE, ROWS, 或 GROUPS的指定如何。

默认RANGE/ROWS行为：

对于大多数窗口函数（如SUM(), AVG(), COUNT()等），如果没有指定RANGE或ROWS，则默认行为通常是RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW，当且仅当包含ORDER BY子句时。

如果没有ORDER BY，则窗口包括分区中的所有行，相当于RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING。

特殊函数的默认行为

RANK()和DENSE\_RANK()等排序相关函数：

如果没有ORDER BY，则所有行将被认为是平等的，并且可能返回同一排名（如1）。

ROW\_NUMBER()函数：

通常需要ORDER BY来生成意义上的行号。如果省略，某些系统可能不允许使用ROW\_NUMBER()，或者给出不确定的行号。

### Hive删除分区语法

ALTER TABLE table\_name DROP [IF EXISTS] PARTITION (partition\_column='xx', );

### 几种窗口函数的区别

ROW\_NUMBER()函数会依次进行排序且序号不相同

RANK()函数会依次排序如结果一样序号相同且会跳跃排名

DENSE\_RANK()会依次排序如结果一样序号相同且不会跳跃排名

Lag 和 Lead函数是SQL中的两个重要的函数，用于查看某一行记录的前一行和后一行数据。Lag函数用于查看上一行数据，而Lead函数用于查看下一行数据。这两个函数可以帮助我们更好地理解某一行记录的上下文关系，并且可以帮助我们做出更准确的分析和决策。

### Apiori

#### 支持度

观测的项集在所有项集中出现的概率

#### 置信度

B在a出现时出现的概率

#### 提升度

A出现对b出现的概率影响；

频繁项集的子集一定频繁

### 随机森林

#### 决策树Cart树

适用于分类问题（Classification Trees）也适用于回归问题（Regression Trees）。

1. 特点

二叉树结构：CART 总是生成二叉树，即每个内部节点分割成两个子节点。

适用于各种数据类型：可以处理数值型数据和类别数据。

使用基尼指数或平均方差最小化进行分割：

在分类问题中，CART 一般使用基尼指数来选择最优的分割点，即使是连续型变量。

在回归问题中，使用平均方差（mean squared error, MSE）最小化来决定分割点。

具有剪枝策略：

使用成本复杂度剪枝（Cost complexity pruning）减少过拟合风险。

其目标是在树的复杂度（通常由树中的节点数表示）和树的性能（通过训练集误差估计）之间找到一个好的平衡。

误差减少剪枝（Error Reduction Pruning）

这是另一种常见的后剪枝技术，直接基于剪枝是否能减少预测误差来进行决策。

#### 随机森林理解

适合大小样本 大样本更好 小样本注意调参

随机性

随机性体现在两个方面：

自助采样（Bootstrap sampling）：每棵树的训练数据都是通过从原始数据集中进行有放回抽样得到的，这意味着一些观测可能会在同一棵树的训练数据中被重复选择，而有些则可能完全不被选择。

随机特征选择：在决策树的每个分裂过程中，候选特征的数量是随机选定的。这不仅减少了计算量，也帮助增加了模型的多样性，从而减少过拟合。

集成

通过集成多棵树，随机森林能够减少模型的方差，不像单一的决策树那样容易过拟合。集成方法通常使用简单投票（对于分类任务）或平均法（对于回归任务）。

集成

通过集成多棵树，随机森林能够减少模型的方差，不像单一的决策树那样容易过拟合。集成方法通常使用简单投票（对于分类任务）或平均法（对于回归任务）。

构建过程

随机森林的构建过程包括以下几个步骤：

创建多个决策树：

对每棵树，随机从原始数据集中抽取N个样本，采用有放回抽样（Bootstrap）。

每次树的分裂考虑的特征是所有特征的一个随机子集。

训练：

每棵树独立地进行训练。

不进行剪枝处理，允许每棵树尽可能深地生长。

预测：

分类任务：每棵树给出一个预测结果，最终输出为多数树选择的类别。

回归任务：每棵树预测一个数值，最终输出为所有树预测值的平均。

优点

高准确性：随机森林在很多情况下都能提供高度准确的预测结果。

防止过拟合：通过集成多棵树，可以显著减少过拟合的风险。

可处理大量特征和分类：适用于有大量输入变量的数据集。

提供特征重要性指标：能够评估哪些特征在预测中起主要作用，对于理解数据非常有用。

缺点

模型解释性差：相比于单棵决策树，随机森林由于包含多棵树，更难以解释。

计算量较大：训练多棵树需要较大的计算资源，尤其是当树的数量和深度较大时。

在某些噪音较大的分类或回归问题上可能表现不佳：特别是对于有大量离群点的数据集。

### Svm

核心目标是最大化分类边界的“间隔”，寻找的两个决策边界离的最远，这俩决策边界叫做支持向量，

优点：

能应用于非线性可分的情况

最后分类时由支持向量决定，复杂度取决于支持向量的数目而不是样本空间的维度，避免了维度灾难

具有鲁棒性：因为只使用少量支持向量，抓住关键样本，剔除冗余样本

高维低样本下性能好，如文本分类

缺点：

模型训练复杂度高：在训练SVM模型时，需要确定支持向量，即距离分类超平面最近的那些样本点。为了找到这些支持向量，需要遍历整个训练数据集，并且需要对每个样本点进行计算，以确定其到超平面的距离。

难以适应多分类问题

核函数选择没有较好的方法论

### 四、选择 SVM 还是随机森林？

选择随机森林：

数据量大，训练速度和性能很重要。

数据集包含许多噪声特征。

需要解释模型的重要性和决策路径。

问题是多分类问题，且不希望进行额外的多类扩展。

选择 SVM：

数据集较小，且分布明确。

问题是二分类问题，并且边界非常清晰。

需要找到精准的决策边界，特别是在处理线性或近线性可分问题时。

适合应用在文本分类、图像识别等特定领域，特别是高维度的数据集上（通过核函数）。

总结

随机森林 是一个强大的、易用的模型，尤其适合数据规模大、特征多、噪声较多的情况。

SVM 适合更小、更干净的数据集，特别是在寻找明确的分类边界时表现优异。

Pca

因子分析

层次分析

逻辑回归

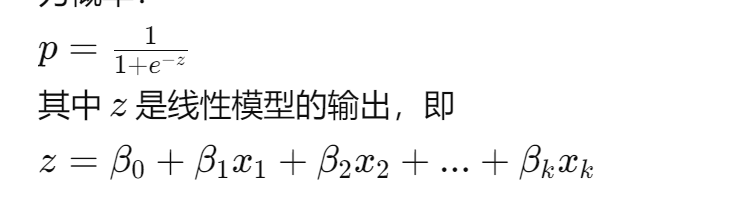
### 条件logit

#### 逻辑回归

是自变量对对数几率函数 进行线性拟合的模型，具体形式

βXT=logit(p)

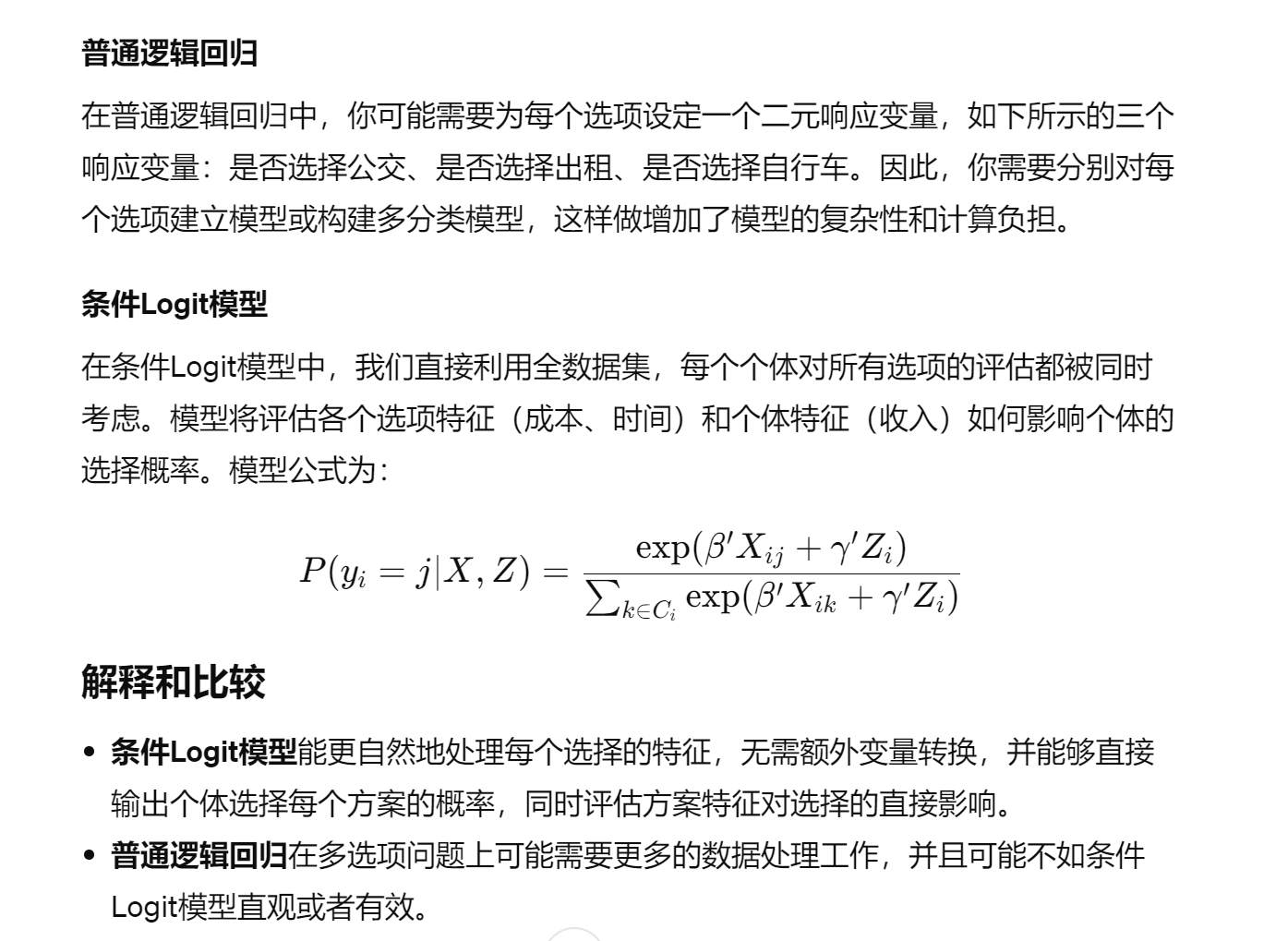
通常会将这个式子变形，把线性函数转换为概率



得到常见的逻辑回归，逻辑回归通常会设定一个阈值 >0.5则分类为1 否则为0

#### 条件logit模型是从logit模型推广而来

条件Logit模型通过引入对每个选项得分的计算，并将这些得分用于概率比较，有效地扩展了Logit模型的应用范围和深度。



如果用普通的logit模型 会导致不同组的数据共用的是同一个随机扰动项，无法控制不同组的这种选择性偏差

检验模型有效性 似然比检验：放入x和不放入x模型是否一样 拒绝则认为模型有效

### 假设检验

控制弃真错误时，原假设究竟怎么设计

1. 疑罪从无，错抓好人是更为严重的结果，默认是好人

原假设：是好人，备择假设：不是好人

1. 错杀一千也不可放过一人，错放坏人是更为严重的结果，默认是坏人

原假设：是坏人，备择假设：不是坏人

说一下显著性水平、置信区间、假设检验？

2.答案：显著性水平：其实就是第一类错误也叫弃真错误，也即原假设为真时被拒绝的概率

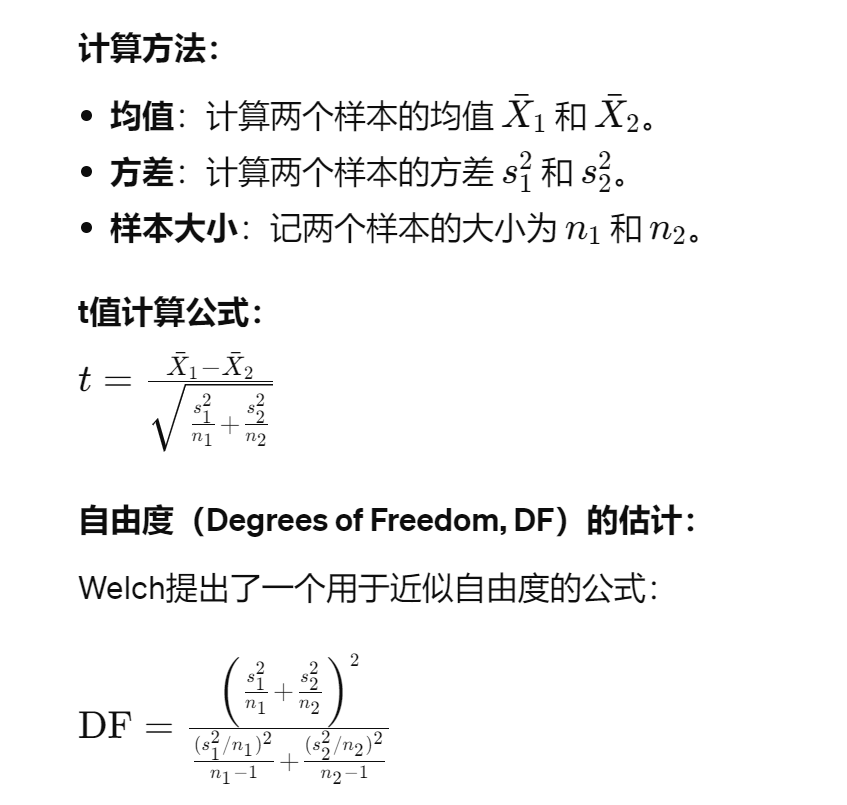
置信区间：在区间估计中，由样本统计量所构造的总体参数的估计区间称为置信区间

假设检验：是一种对预先提出的假设进行检验的一个框架，目的是通过样本信息验证对整体的假设是否成立。

均值——t检验 构造统计量 n \* x-u/s2，

做t检验的时候需要先做F检验检验两样本方差是否相等；

做t检验如果方差不相等Welch's t-test 韦尔奇t检验



方差：方差通常检验比值 单样本的方差 因为比的是已知值因此构造 卡方分布用卡方检验X（n-1）；两个样本由于是两个样本方差比因此构造F分布F（m-1，n-1）方差： 单样本方差 用卡方 两样本用F

比率P 检验：

通过F分布检验，我们可以得知整体模型的拟合程度，但无法确定具体哪些系数不为零。如果需要确定具体系数的显著性，通常需要进行t检验或者Z检验。

图片包含 文本

描述已自动生成

卡方检验：列联表独立性，拟合优度检验

## Rfm模型和生命周期模型

R：最近一次消费时间

F：消费频率

M：消费金额

带重要的都是 M是80%以上

重要保持：RFM均大于80分位数

重要价值：最近消费不满80

重要发展：频次和最近消费不满80

重要挽留：频次不满80

非一般：M和F大于99.5

### 生命周期模型

注册时间 30和90是两个节点

最近一次消费时间90 180 365 是节点

注册30天是新人期

### 访谈部分：

用户侧写：怎么定义有价值的信息，有价值的信息是什么样的

### 数据治理：

会员id 下单时间，商品信息 一一比对

### 门店评分：

因子分析原理，层次分析

### 如何避免决策树过拟合

限制树深

剪枝

限制叶节点数量

正则化项

增加数据

bagging（subsample、subfeature、低维空间投影）

数据增强（加入有杂质的数据）

早停

04

### 朴素贝叶斯的理解

理解： 朴素贝叶斯是在已知一些先验概率的情况下，由果索因的一种方法

其它： 朴素的意思是假设了事件相互独立

05

## 业务场景题

### 幸存者偏差（Survivorship Bias）

是一种常见的统计偏差，特别是在A/B测试和其他实验设计中，如果不加以注意和处理，会导致误导性的结论。了解和识别幸存者偏差在A/B测试中的影响是至关重要的，因为它可以显著影响实验结果的解释和决策过程。以下是幸存者偏差的介绍和它在A/B测试中的具体意义。

幸存者偏差定义

幸存者偏差是指在分析或决策过程中，只考虑那些“幸存”或通过某个筛选过程的个体或事例，而忽略了那些没有幸存的个体或事例。在A/B测试的上下文中，这种偏差可能发生如果数据从未完整记录或者在分析时未将所有参与者纳入考量。

A/B测试中的幸存者偏差

在A/B测试中，幸存者偏差可能以多种形式出现，下面是一些示例：

1. 早期退出问题

问题描述：在一个产品改进的A/B测试中，可能只有那些使用产品足够长时间的用户才显示出某种效果（比如改进的用户满意度）。如果那些因为不满意而早早退出的用户被忽略，那么结果可能会偏向那些“幸存”的用户。

意义：这可能导致高估产品改进的正面效果，因为不满意的用户（可能因为新功能而离开）的数据没有被包括在内。

2. 高参与度用户偏差

问题描述：在测试新的网站功能时，可能只有最活跃的用户会使用到这些新功能，而偶尔访问的用户则可能完全没有触达到新功能。

意义：分析可能错误地表明新功能非常成功，仅仅因为它只考虑了那些已经是活跃用户的数据，而这部分用户可能本来就更可能给出正面反馈。

如何应对幸存者偏差

处理幸存者偏差的关键在于确保所有相关的数据样本都被纳入分析，不论其结果如何。以下是一些应对策略：

全面数据收集：确保从实验开始到结束，所有用户的数据都被完整记录，包括那些退出测试的用户。

分层分析：对不同类型的用户（如按活跃度分层）进行分层分析，确保理解各个子群体的反应。

### 辛普森悖论

辛普森悖论：

x对y的影响 当对x进行分组时 得到的结论可能会跟x总体跟y的关系完全相反，这个组就是混杂变量，混淆因子对结论的影响

### 如何分析次日留存率下降的问题？

业务问题关键是问对问题，然后才是拆解问题去解决。

（1）确认数据以及统计来源的准确性；

（2）了解指标对应的具体业务情况以及异常情况，判断是预期内的正常变动还是预期外的异常情况；

（3)将数据指标进行拆解；

（4）内外部影响因素分析；

（5）预测未来是否还会异常？应该采取什么措施避免异常再次发生？

（6）与业务沟通反馈分析结论，探讨后续方案的执行。再针对原因解决问题，制定优化策略。

**1. 两层模型**

从用户画像、渠道、产品、行为环节等角度细分，明确到底是哪里的次日留存率下降了

**2. 指标拆解**

次日留存率 = Σ 次日留存数 / 今日获客人数

**3. 原因分析**

**内部：**

* 运营活动
* 产品变动
* 技术故障
* 设计漏洞（如产生可以撸羊毛的设计）

**外部：**

* 竞品
* 用户偏好
* 节假日
* 社会事件（如产生舆论）

### 处理需求时的一般思路是什么

（需要举例）

* 明确需求，需求方的目的是什么
* 拆解任务
* 制定可执行方案
* 推进
* 验收

# 机器学习

## 特征工程怎么做的，选择了哪些特征作为预测变量？为什么用RFM模型来构建特征变量？

是指用一系列工程化的方式从原始数据中筛选出更好的数据特征，以提升模型的训练效果。

包括：数据预处理、特征选择、降维等环节，也可以说是特征构建、特征提取、特征选择。

特征选择：

过滤式：计算特征与目标列的相关系数、通过卡方检验计算每列与目标列的相关性，缺点未考虑特征间相关性

包裹式：通过随机森林对特征重要性进行分析与排序、logistics回归

特征工程包括：特征构建->特征提取->特征选择。 选择特征：用户行为特征、用户消费特征、用户画像特征

## 为什么RFM模型：

因为我们没有太多的用户行为数据，能用的数据比较有限。但是有一定的成交数据。只要有成交数据，就能进行RFM的分析。其次，模型的分层可解释性强。其他很多算法模型、机器学习模型，往往通过聚类进行用户的分层，对于业务来讲，不是很好解释。但RFM模型是非常好理解的。

## 有关机器学习random forest 和xgboost的区别？

Id3，C4.5,CART、随机森林

C4.5信息增益比，悲观剪枝，可处理连续型

Cart基尼系数，代价复杂度剪纸，可回归，可处理连续性

随机森林，baging和boosting方法多次生成树投票决定，一般是以cart为基学习器

答案：RF：采用Bootstrap的随机有放回的抽样，抽样出N份数据集，训练出N个决策树。然后根据N个决策树输出的结果决定最终结果。 xgboost：通过boosting的思想，由一堆cart树，将每棵树的预测值加在一起就得到了最后的预测值。 RF属于集成学习Bagging，而XGB属于集成学习Boosting。 RF是通过减少模型方差提高性能；XGB是通过减少模型偏差提高性能。 对于最终的输出结果而言，RF采用多数投票等；而XGB则是将所有结果累加起来，或者加权累加起来。

采用多数投票等；而xgboost则是将所有结果累加起来，或者加权累加起来。

4、随机森林对异常值不敏感；xgboost对异常值非常敏感。

5、随机森林对训练集一视同仁；xgboost是基于权值的弱分类器的集成。 答案解析 这道题主要是考察xgboost和随机森林的区别，只要能大致描述几个本质上的区别即可

7.L1、L2的原理？两者区别？

答案：原理： L1正则是基于L1范数和项，即参数的绝对值和参数的积项；L2正则是基于L2范数，即在目标函数后面加上参数的平方和与参数的积项。 区别：

1.鲁棒性：L1对异常点不敏感，L2对异常点有放大效果。

2.稳定性：对于新数据的调整，L1变动很大，L2整体变动不大。

9.假设检验的原理和步骤？

答案：假设检验的原理： 小概率事件原理，小概率事件在一次实验中基本是不可能发生的，而一旦发生就有充分的理由拒绝原假设。去证明假设是错误的，从而反证假设的另一面很可能是正确的，运用的是反证法。 假设检验的步骤： 确定原假设和备择假设 确定适当的检验统计量，并计算其数值。选择哪个统计量作为检验统计量需要考虑一些因素， 最后看这个数据是落在接受域还是拒绝域，如果落在接受域则接受原假设，如果落在拒绝域则接受备择假设 。

10.你对PCA了解吗？

答案：在统计学中，主成分分析(PCA)是一种简化数据集的技术。它是一个线性变换。这个变换把数据变换到一个新的坐标系统中，使得任何数据投影的第一大方差在第一个坐标(称为第一主成分)上，第二大方差在第二个坐标(第二主成分)上，依次类推。主成分分析经常用减少数据集的维数，同时保持数据集的对方差贡献最大的特征。这是通过保留低阶主成分，忽略高阶主成分做到的。这样低阶成分往往能够保留住数据的最重要方面。但是，这也不是一定的，要视具体应用而定。 PCA的算法步骤如下: 设有m条n维数据。

1、将原始数据按列组成n行m列矩阵X

2、将X的每一行(代表属性字段)零均值化，即减去这一行的均值

3、求出协方差矩阵

4、求出协方差矩阵的特征值及对应的特征向量

5、将特征向量按对应特征值大小从上到下按行排列成矩阵，取前k行组成矩阵P Y=PX即为降维到k维后的数据

不一致：多个数据库对比

重复：删除

不完整：删除或填充插值、热卡、k近邻、回归

含噪声：

1）通过一定的算法识别出离群点。一般是使用该数据标准差的多少倍来判断。比如正太分布中，正负标准差3倍以上的概率是99.7%，可以将其认定为可疑离群点。

2）人工判断离群点是属于错误数据导致的，还是正常的离群点。

3）对离群点进行处理。一般来说，错误的离群点需要更正或者删除。正常的离群点则需要另外建模进行分析。

纬度高：降维

## 模型评价指标

分类指标：

Accuracy：准确率=分类正确样本数/总样本数

Precision：精确率=正确预测的正例数 /预测正例总数

Recall：召回率=正确预测的正例数 /实际正例总数

F1 score：F1分数=Precision与Recall的调和，既考虑查准也考虑查全

AUC：横坐标为FPR，纵坐标为TPR的曲线(AUC曲线)下面积

回归指标：

SSE：残差平方和 MSE：均方误差 RMSE：均方根误差 R方：拟合优度 MAE：平均绝对误差

聚类指标： 轮廓系数 手肘法确定聚类数

# 简历项目部分

1. 业务理解与分析： 你在百联全渠道电子商务有限公司的工作经验中提到参与了会员画像和消费行为分析，还使用了RFM和生命周期模型。请具体谈谈你在这些项目中的贡献和如何使用这些分析帮助业务决策？

贡献：1、数据预处理，将会员信息和订单信息关联，搭建分析用的会员订单信息表，计算标签如RFM 、生命周期、新客老客 黄牛标签

2、分层分析，新客喜欢买热度高的商品，热度高的商品；rfm分层熟客买阿迪耐克，新客买食品

3、黄金珠宝类获客比例高

4、线上消费者比线下消费买的东西更便宜，t检验

并通过 RFM 模型对会员进行了细致的划分。RFM 模型主要通过最近一次消费时间（Recency）、消费频率（Frequency）、以及消费金额（Monetary）三大维度来评估客户的价值。我对公司所有会员进行了分类，识别出了高价值客户、潜力客户和即将流失的客户群体。之后，通过生命周期模型，我分析了不同阶段客户的行为特征，构建了个性化的营销方案，帮助市场团队进行精准营销。

如何帮助业务决策： 通过这些分析结果，我们能够更好地分配营销资源。例如：

针对高价值客户：我们推出了 VIP 活动、专属折扣等个性化服务，提升了客户的忠诚度，购物抽金条

针对新客：提供定向促销活动，刺激二次消费，推动了更多会员转化为高价值客户。 多波段营销

2. 数据处理与技术能力： 你提到在数据挖掘与建模方面，搭建了门店评分体系，并使用了AHP层次分析法。能否详细说明你是如何选择和处理数据的？这些模型在实践中如何提高了决策的准确性？

数据处理与技术能力：在使用 AHP（层次分析法）搭建门店评分体系时，首先我们需要

确定特征，考虑门店经营情况和数字化情况，四个角度：

1、基本经营情况；2、经营质量；3、数字化运营，券、cdp、BI使用；4、数字化服务无感积分、智能停车

AHP打分确定权重

时间上滚动一年，并进行了数据清洗，对空值填补，剔除缺失值，对极端值采用分位数归一化

帮助更合理地预测门店的综合表现

从而提升管理层在资源分配和绩效评估上的决策准确性。

3. 报表和BI工具： 在你的工作中，你从0到1搭建了自助式即席查询报表，并通过SQL调度优化了多个BI看板。请详细描述你在搭建报表过程中遇到的挑战，以及你是如何通过SQL和BI工具解决这些挑战的？

挑战主要体现在如何高效处理大规模数据和确保查询的灵活性。

为此，我优化了 SQL 查询调度，通过分区提高查询速度。

在搭建 BI 看板时，用户可以自定义筛选条件以获得实时数据分析

4. 数据挖掘项目经验： 你在实习中提到了量化研究以及期货涨跌趋势分析。可以分享一下在分析过程中，你遇到了哪些数据处理上的挑战？你是如何利用Python工具来应对这些挑战的？

在期货涨跌趋势分析中，数据量庞大且波动频繁，噪音干扰较多。

## 问题1：CDP会员人群标签建设

你在简历中提到了为CDP会员人群标签合理调整表结构以减少运算压力。能不能详细解释一下，你是如何分析并优化表结构的？这个过程如何提升系统的效率？

优化主要体现在两个方面一个是结果表，数据量的优化，之前的标签大量使用union 导致表没有主键，修改之后使用full outer join 增加字段，降低了结果表的数据量

小文件的优化，中间结果表大量使用union all spark引擎在做union all操作的时候会产生大量小文件，使用distribute by 语法将数据存入一个hdfs文件中

问题2：排行榜系统

在你推动的i百联app联华超市外卖板块的排行榜项目中，你负责1000+门店和15万+商品的数据得分、排名计算。请详细谈谈你是如何设计这些评分与排名算法的

好评榜，复购榜，热卖榜

好评榜：贝叶斯平均 先验好评数C 先验好评率r 实际好评数m 实际好评率n (C\*r+m\*n)/C+m

问题3：数据治理

你提到主动发现并处理了多个重大数据问题。能否分享一个具体的案例，描述问题的发现过程、沟通的细节以及你是如何协调内外部团队一起解决这个问题的？

我们哈啰出行的数据治理中，涉及多维度的出行数据和复杂的系统整合，遇到数据异常或误差时，你会如何快速定位问题的根源，并确保系统的准确性？

1、数据完整性监控，数据范围或阈值监控

可以定期检查表中的记录是否符合预期的数据量，是否有遗漏或冗余数据。

2. 数据一致性验证

通过SQL检查不同表或字段之间是否有不一致的数据，比如外键关系、数据格式等。

SELECT \*

FROM child\_table c

LEFT JOIN parent\_table p ON c.parent\_id = p.id

WHERE p.id IS NULL;

这个查询检查子表child\_table中是否存在没有匹配父表parent\_table的记录，确保外键关系正确。

3. 数据重复性监控

监控是否有重复数据，例如重复的订单号、用户ID等，这类监控可以帮助你检测到数据插入时的重复问题。

问题4：跨部门协作

你在项目中有过很多跨部门合作的经验。哈啰出行的业务较为复杂，涉及到产品、运营、技术等多个部门。能否分享一次你在项目中遇到的跨部门沟通的挑战？你是如何处理的？在哈啰出行这样一个快速发展的企业，你会如何确保沟通顺畅、项目顺利推进？

因为前端、后端、运营团队的需求不同，我需要协调这些部门来确保项目顺利进行。

为此，我经常主持跨部门会议，明确每个团队的需求和痛点，确保大家朝着共同的目标推进。