# 向上科技 爬虫实习生

python 数据类型 数据结构 线程进程

## 1. 数据类型

数据类型是指在编程中用于表示数据的种类。Python 中常见的数据类型包括：

整数（int）：表示整数值，例如 5、-10。

浮点数（float）：表示带小数点的数值，例如 3.14、-0.001。

字符串（str）：表示文本数据，例如 "Hello, World!"。

布尔值（bool）：表示真（True）或假（False）。

列表（list）：用于存储有序的可变集合，例如 [1, 2, 3]。

元组（tuple）：用于存储有序的不可变集合，例如 (1, 2, 3)。

字典（dict）：用于存储键值对，例如 {'name': 'Alice', 'age': 30}。

集合（set）：用于存储无序的唯一元素，例如 {1, 2, 3}。

## **2. 数据结构**

数据结构是组织和存储数据的方式，以便能够高效地进行访问和修改。Python 中常用的数据结构包括：

列表（List）：动态数组，支持索引、切片、添加、删除等操作。

字典（Dictionary）：键值对集合，提供快速的查找和插入。

元组（Tuple）：不可变序列，适合存储不需要修改的数据。

集合（Set）：无序集合，支持集合操作，如并集、交集等。

链表（Linked List）、栈（Stack）、队列（Queue）：这些数据结构通常需要通过类和对象来实现。

## **3. 线程**

线程是进程中的一个执行单元，多个线程可以在同一个进程内并发执行。Python 中的线程主要通过 threading 模块实现。

优点：

线程共享进程内的资源，如内存和文件句柄。

适合处理 I/O 密集型任务，如网络请求和文件操作。

注意：

Python 的全局解释器锁（GIL）限制了多线程在 CPU 密集型任务中的效率。

## **4. 进程**

进程是操作系统分配资源的基本单位，每个进程有自己的内存空间和资源。Python 的 multiprocessing 模块允许创建多个进程，以实现并行计算。

优点：

每个进程有独立的内存空间，不受 GIL 的限制，适合 CPU 密集型任务。

提高程序的运行效率。

注意：

进程间通信（IPC）比线程间通信更复杂，因为它们有独立的内存空间。

## **sql 删除 更新**

DELETE FROM 表名 WHERE 条件;

UPDATE 表名 SET 列名 = 新值 WHERE 条件;

计算机网络

## **Sql的索引**

1. 主键索引（Primary Key Index）

主键索引是唯一的索引，用于唯一标识表中的每一行。默认情况下，主键索引是聚集索引。

2. 唯一索引（Unique Index）

唯一索引确保索引列的值唯一，不允许重复。可以用于任何列，除了主键列。

3. 聚集索引（Clustered Index）

聚集索引决定了数据行在表中的存储顺序。每个表只能有一个聚集索引，因为数据只能以一种顺序存储。

4. 非聚集索引（Non-Clustered Index）

非聚集索引存储了索引键值和指向数据行的指针。一个表可以有多个非聚集索引。

5. 全文索引（Full-Text Index）

全文索引用于对文本列进行高效的全文搜索，适用于大文本数据的搜索。

6. 复合索引（Composite Index）

复合索引是基于多个列创建的索引，适合于经常用于 WHERE 子句中组合查询的多列。

7. 位图索引（Bitmap Index）

位图索引使用位图来表示列中每个值的存在情况，适用于低基数（即列中不同值较少）的情况，常用于数据仓库。

8. 哈希索引（Hash Index）

哈希索引用于基于哈希表实现的索引，适用于等值查询，但不支持范围查询。

9. 空间索引（Spatial Index）

空间索引用于优化地理数据的查询，适合于存储和查询空间数据类型（如地理位置）。

10. 临时索引（Temporary Index）

临时索引在查询过程中动态创建，常用于复杂查询的优化，但在会话结束后会被删除。

## **sql筛选特定的日期类型**

SELECT \* FROM sales

WHERE sale\_date LIKE '\_\_\_\_-\_\_-\_\_'; -- 4位年份、2位月份、2位日期

在这里，\_ 表示单个字符，因此 \_\_\_\_-\_\_-\_\_ 确保有四位数字、两位数字和两位数字的准确格式。

## **栈的基本操作时间复杂度：**

入栈：O(1)

出栈：O(1)

查看栈顶元素：O(1)

判断是否为空：O(1)

队列的基本操作时间复杂度：

入队：O(1)

出队：O(1)

查看队头元素：O(1)

判断是否为空：O(1)

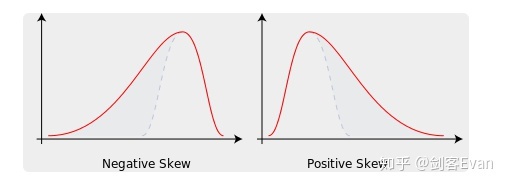
## **Python的类**

## **Pandas转换成dataflow**

# 中软国际数据治理实习生

## 偏度处理

偏度处理也是异常点处理，异常点会把数据的分布拉偏，这就好比二八法则，会把数据拉偏。



## **判断异常点来源**

判断异常点的可靠性(错误或失误导致)

异常点和正常点是否来源于不同的生成机制

**处理方式**

异常点移除(总的数据量多的时候)

空间标识先对同一样本的所有特征进行标准化，然后再单位化,这样处理可以把数据拉到高纬度空的球面上。

去对数(某一特征)

求平方根(某一特征)

求倒数(某一特征)

## **缺失值处理**

使用统计数据填充

使用算法填充 -K紧邻-线性回归

## **移除不重要的特征**

PCA(主成分分析)

移除不具有区分度的特征

该特征的某个特征的特征值出现的最高频率是次高频率的20倍以上(可删除)

# 字节机器学习数据科学家-经营办公室

一面

1、自我介绍

2、机器学习中常见的回归和分类算法

## 回归算法

### 线性回归（Linear Regression）

假设自变量与因变量之间的线性关系。

简单易理解，适合处理线性关系。

### 岭回归（Ridge Regression）

在线性回归的基础上加入L2正则化，防止过拟合。

适合特征较多的高维数据。

### Lasso回归（Lasso Regression）

在线性回归的基础上加入L1正则化，有助于特征选择。

可以将某些特征的系数压缩为零。

对不太重要的特征权重进行压缩：由于L1正则化项中的绝对值操作，当某个权重 wi 对整体损失函数影响较小时，优化算法倾向于将其变为零，以最小化正则化项。

### 决策树回归（Decision Tree Regression）

使用决策树模型进行回归，适合**处理非线性**数据。

易于解释，但容易过拟合。

首先，我们假设有类别 label，计算不纯度的度量（如 D）。然后，我们分别计算各个特征之后，系统的纯度。我们可以计算每个特征的信息增益（Gain(D) = H(D) - H(D | A)），并评估信息增益是否足够高来进行分裂。

### 随机森林回归（Random Forest Regression）

使用 Bagging 算法，即有放回采样地选取 n 个样本，建立 m 个决策树分类器。多个分类器采用**投票机制**来产生最终的分类结果。

使用多个决策树的集成方法，提升预测性能。

对于大多数数据集具有较好的泛化能力。

### 支持向量回归（Support Vector Regression, SVR）

基于支持向量机的回归方法，适合**处理高维数据**。

可以通过核函数处理非线性关系。

### 梯度提升回归（Gradient Boosting Regression）

通过逐步构建决策树，提高模型的预测能力。

适合处理复杂的非线性关系。

### XGBoost

XGBoost（Extreme Gradient Boosting）是一种基于梯度提升树（Gradient Boosting Trees）的方法，广泛应用于分类和回归任务

当每个弱分类器的分类效果不理想时，我们可以尝试将多个不同的弱分类器组合起来，形成效果更好的强分类器。Boosting 着重于减小模型的 bias。

**为何说 Bagging 是减小 Variance，Boosting 是减小 bias?**

Bagging

模型多样性：由于每个模型在不同的训练子集上训练，它们会学习到不同的数据特征，导致模型之间的预测结果有一定的差异。

误差抵消：由于模型的预测错误是随机的，多个模型的组合可以通过平均化来抵消个别模型的错误，从而降低整体的方差。

Boosting 通过逐步训练多个模型，每个新模型都试图纠正前一个模型的错误。

## 分类算法

### 逻辑回归（Logistic Regression）

**逻辑回归主要用于分类问题，对于所给数据集，认为可以用一条直线将数据线性分类。**输出为概率值。

可扩展到多分类问题（如多项式逻辑回归）。

逻辑回归（Logistic Regression）是一种用于二分类问题的统计模型，尽管名称中有“回归”二字，但它主要用于分类。逻辑回归通过将线性回归的输出映射到一个概率值，来进行分类决策。以下是逻辑回归的详细原理介绍。

**1. 基本概念**

目标：逻辑回归的目标是预测某个事件发生的概率，通常是二分类问题（例如，判断某个邮件是否为垃圾邮件）。

输出范围：逻辑回归的输出是介于0和1之间的概率值，表示样本属于某一类别的可能性。

**2. 工作原理**

**1. 线性组合**

逻辑回归首先计算输入特征的线性组合：



其中：

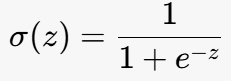
w 是特征的权重向量。

x 是输入特征向量。

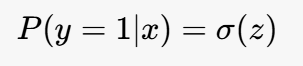
b 是偏置项（截距）。

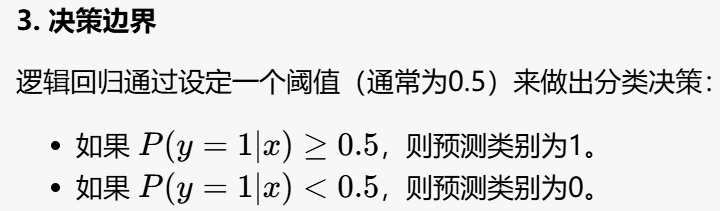
2. Sigmoid函数

为了将线性组合的结果



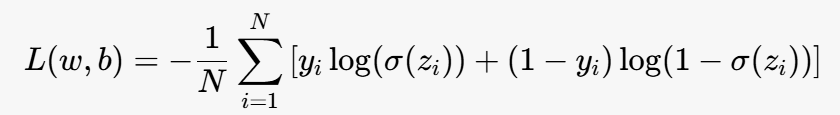
z 转换为概率值，逻辑回归使用Sigmoid函数（也称为逻辑函数）：

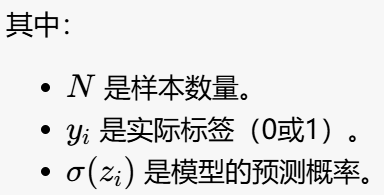


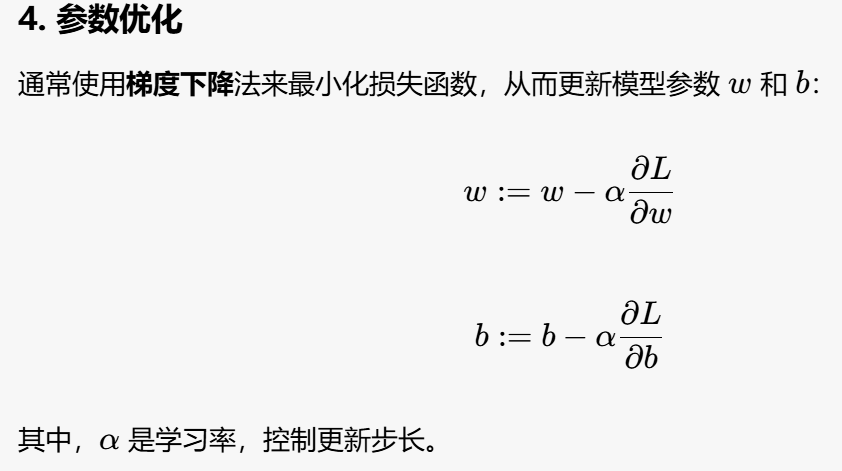


**3. 损失函数**

为了训练逻辑回归模型，需要优化模型参数 w 和 b。使用的损失函数为对数损失（Log Loss）：







### 决策树分类（Decision Tree Classifier）

使用树形结构进行分类，直观易解释。

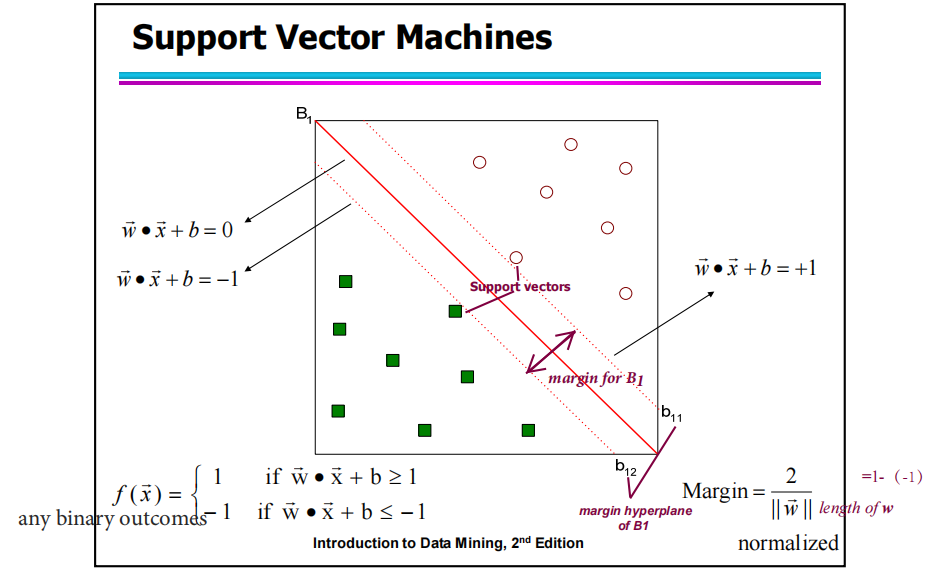
易受噪声影响，容易过拟合。

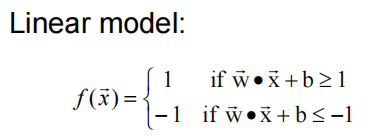
### 随机森林分类（Random Forest Classifier）

使用多个决策树的集成方法，提高分类准确性。

对于大多数数据集具有较好的表现。

### 支持向量机（Support Vector Machine, SVM）





通过找到最佳分隔超平面进行分类，适合处理高维数据。

可以处理非线性分类问题（通过核函数）。

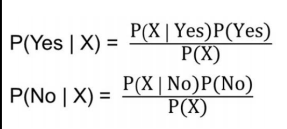
### K近邻（K-Nearest Neighbors, KNN）

基于距离度量进行分类，简单易懂。

对于大数据集计算开销较大。

### 朴素贝叶斯分类（Naive Bayes Classifier）

P(Yes|X) = P(X|yes)\*P(Yes)



基于贝叶斯定理，假设特征条件独立。

适合文本分类等任务。

#### 朴素贝叶斯分类器的优缺点

朴素贝叶斯分类器的**主要优点**有

模型发源于古典数学概论，算法比较简单，且有稳定的分类效率:

对小规模的数据表现好，适合多分类任务

对缺失数据不敏感，例如文本分类等任务

不存在过拟合的说法。

朴素贝叶斯分类器的**缺点**有:

朴素贝叶斯假设特征之间相互独立，但在现实中这个假设往往不成立

朴素贝叶斯需要先估计先验概率，如果估计不准确容易影响分类结果

基于概率的分类有可能会不准确，

### 梯度提升分类（Gradient Boosting Classifier）

**1. 初始化模型**

从一个简单的模型开始，通常是所有样本的平均值（对于分类问题，可能是出现频率最高的类别）。

**2. 计算残差**

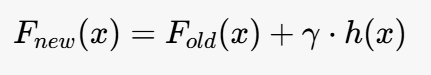
对于每个样本，计算当前模型的预测值与实际值之间的差异，这个差异称为残差。残差表示模型的预测误差。

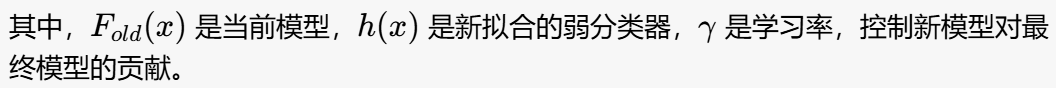
**3. 拟合残差**

使用一个新的弱分类器（通常是一棵决策树）来拟合残差。这个新模型的目标是尽量减少当前模型的预测误差。

**4. 更新模型**

将新模型的预测结果与当前模型的预测结果结合，以更新模型。更新公式为：





**5. 重复迭代**

重复步骤 2 到 4，直到达到预设的迭代次数或模型的性能不再显著提高为止。

通过逐步构建弱分类器（如决策树）进行集成，提升分类性能。

通常可以达到非常高的准确率。

## 聚类模型

### K均值聚类（K-Means Clustering）：

随机选择K个初始中心点，迭代分配每个样本到最近的中心点，然后更新中心点，直到收敛。

### 层次聚类（Hierarchical Clustering）：

通过逐步合并或分割数据点，生成树状结构（树状图），可以选择不同的层次进行分组。

### DBSCAN（Density-Based Spatial Clustering of Applications with Noise）：

根据数据点的密度进行聚类，能够识别任意形状的簇，并处理噪声点。

### Gaussian Mixture Models（GMM）：

假设数据由多个高斯分布组成，通过EM算法估计每个分布的参数，并进行聚类。

## 4. 关联规则学习

### Apriori算法

通过频繁项集的生成，计算项集的支持度和置信度，从而发现变量之间的关联规则。

### Eclat算法

采用深度优先搜索来发现频繁项集，利用事务ID进行高效计算。

## 5. 深度学习模型

**为什么神经网络要偏置项？**

### 对于神经网络中的每一个神经元，都有：y = WT \* x + b。这个式子就是要用来控制神经元的输出决策。而**结果有时会偏离理想状态**，那么就必须有偏置项来修正这个偏差。偏置项加在输入的线性组合上能够让模型更灵活。如果没有偏置项，就可能导致模型发生 bug。

**在所有场合都可以使用偏置项吗？**

不是。例如在卷积层之后，如果要添加 Batch Normalization 层，最好不要添加偏置项，因为不再起作用。在 BN 中，有一步是标准化：

x̂\_i = (x\_i - μ) / σ^2 + ε

其中，μ 为均值，σ^2 为方差，在该操作中，偏置项会被计算掉，故偏置项不再起作用。

3.梯度消失和梯度爆炸问题

在反向传播的梯度更新中，若更新的梯度一直小于 0，就可能触发连乘效应，在之后的传播中越传越小导致靠近输入层的权值几乎不更新，训练收敛速度变慢，这便是梯度消失。与之相反，若梯度过大则会触发梯度爆炸，以致于溢出，出现梯度为 NaN 的问题。

### 常见的优化器及对比

| 方式 | 特点 |
| --- | --- |
| GD / SGD / MBGD | 1.难以选择合适的学习率。学习率太小会导致收敛慢，学习率太大会导致波动过大，可能跳出局部最优解。  2.每个参数的学习率都是相同的。如果数据是稀疏的，且不同特征的出现频率相差较大，则会出现部分特征学习不足的问题；  3. 在训练中容易陷入鞍点，即局部最优点，在这些点的梯度为0，无法继续训练。 |
| Momentum | 1.借鉴了物理中的动量概念，模拟物体运动的惯性，即更新的时候在一定程度上保留之前更新的方向，而不是像GD算法一样完全按照新的梯度方向更新。这样可以增加稳定性，并且有一定的摆脱局部最优解的能力。  2. Momentum算法会观察上一步的梯度，当当前梯度方向与历史梯度一致，则增强该方向的梯度，否则则削弱该方向的梯度。 |
| AdaGrad | 1. 针对GD算法中对于每个参数保持同一学习率的问题，AdaGrad算法能在训练中自动对不同参数的学习率进行调整。对于出现频率较低的特征，增大更新的学习率；对于出现频率高的，减小学习率。2. 由于这个特性，AdaGrad非常适合用于处理稀疏的数据。 |
| RMSprop | 1.Root Mean Square prop. 对AdaGrad算法的改进，把AdaGrad的梯度历史梯度相加变成对历史梯度求均值；  2. 这种方法可以缓解AdaGrad算法学习率下降较快的问题。 |
| Adam | 1.Adam算法结合了AdaGrad和RMSprop的优点，动态地更新参数的学习率。不同于RMSprop只参考了参数的历史平均值，Adam同时参考了梯度的均值和方差。  2. 在各大机器学习库中，两次估计的取值率通常分别为β1和β2分别为0.9和0.999。 |
| AdamW | 1. 针对Adam算法中先进行梯度更新再进行正则化，使得梯度大的参数无法被正则化的问题，在AdamW中将梯度衰减的步骤移到正则化之前，解决了这一问题。 |

### **人工神经网络（Artificial Neural Networks, ANN）：**

由输入层、隐藏层和输出层组成，通过激活函数和权重调整进行非线性学习。

### 卷积神经网络（Convolutional Neural Networks, CNN）：

通过卷积层提取图像特征，使用池化层降低维度，适合处理图像数据。

### 循环神经网络（Recurrent Neural Networks, RNN）：

处理序列数据

RNNs 能够处理序列数据，如时间序列、文本、音频和视频，因为它们具有“记忆”的能力，可以保留和使用前面输入的信息。这对于理解上下文和序列依赖关系非常重要。

### 长短期记忆网络（LSTM）

RNN的一种变种，设计了门控机制来控制信息的流动，能够处理长序列依赖问题。

### 生成对抗网络（GAN）

由生成器和判别器组成，生成器生成假数据，判别器判断数据的真伪，二者对抗训练。

3、AB实验：全流程、两类错误、p值、样本量、中心极限定理、Z检验&t检验、卡方检验，绝对值指标以及比值指标各自适用的检验

## Abtest

## A/B 实验原理

### 1. 原理

**实验设计**：

* + 将用户随机分为两个（或多个）组：实验组（A组）和对照组（B组）。A组接收一种版本（如新功能或设计），B组接收另一种版本（如现有功能或设计）。

**指标测量**：

* + 选择关键性能指标（KPI），如点击率、转化率、用户留存等，以评估两组的表现。

**统计分析**：

* + 使用统计方法（如 t 检验或卡方检验）来判断两个组之间的差异是否显著。

### 2. 目的

* 确定哪种版本更有效，以便做出数据驱动的决策。

## A/B 实验的难点

### 1. 样本量

**样本量不足**：样本量过小可能导致统计显著性不足，无法准确评估效果。

**样本量过大**：样本量过大可能导致实验成本高，且不必要的延迟结果。

### 2. 随机化

* **随机分配**：确保用户的随机分配，以避免偏差。非随机分配可能导致结果不可靠。

### 3. 时间与周期

* **实验周期**：确定合适的实验周期，以确保结果不受季节性或其他外部因素的影响。

### 4. 统计显著性

* **多重测试问题**：在进行多个 A/B 测试时，可能会增加假阳性的风险，需要控制显著性水平。

### 5. 用户体验

* **用户影响**：实验可能影响用户体验，例如，频繁的界面变更可能导致用户困惑。

## 流量分层

流量分层是一种将流量根据不同特征或属性进行分配的方法，以确保 A/B 实验的结果更具代表性和可靠性。以下是流量分层的几种常见方法：

### 1. 基于用户特征的分层

**按地域**：根据用户的地理位置，确保不同地区的用户均匀分布在 A/B 测试中。

**按设备类型**：根据用户使用的设备（如手机、平板、桌面）进行分层，确保不同设备的用户均匀分配。

### 2. 基于行为的分层

* **按用户行为**：根据用户的历史行为（如新用户、活跃用户、沉默用户）进行分层，以确保不同类型用户的反馈能够被充分考虑。

### 3. 按流量来源分层

* **按流量来源**：根据用户访问网站的渠道（如自然搜索、社交媒体、广告）进行分层，以确保不同来源的用户均匀分布。

### 实施流量分层的步骤

1. **确定分层标准**：选择合适的分层标准（如地域、设备、行为等）。
2. **收集数据**：在实验前收集基础数据，以便进行分层。
3. **进行随机分配**：在每个层次内进行随机分配，确保每个组的用户特征一致。
4. **监测与分析**：在实验过程中监测各层的表现，并在分析结果时考虑分层。

### 流程

确定目标和假设->确定指标->确定实验单位->计算样本量->实施测试->分析实验结果

其中确定指标中比较关键的是要确定评价指标和护栏指标，评价指标就是驱动公司实现核心价值的指标，要具有可归因性、可测量性、敏感性和稳定性；护栏指标也就是辅助指标

● 确定实验单位有从用户层面、访问层面和页面层面进行考虑的情况，用户层面适用于易被用户察觉的变化实验，访问和页面层面适用于不易被用户察觉的变化实验；从用户层面到页面层面实验粒度越来越细，累计的样本量也越来越多

● 计算样本量，需要预先确认以下数值：显著性水平、功效、实验组和对照组的综合方差以及期望的最小差值。实验组和对照组数据量最好均分，非均分的时候只有相对较小的组达到最小样本量，实验结果才可能显著，并不是说实验组越大越好，因为瓶颈是在样本量较小的对照组上，所以实验组和对照组的样本量最好相同

● 分析测试结果的时候要注意辛普森悖论等问题，而且要保证样本达到足够的量、检验是否在正常的波动范围内

### AB测试在什么平台上进行？介绍一下主要步骤？（字节跳动）

参考答案

（1）定义策略：确定分流的目的、放量规模、递增的频率、回滚的策略等；

（2）筛选用户：确定分流访问的用户特征，定义规则（根据IP，user\_id，cookie，业务需求（商户）等因素，指定分流策略）或导入名单；

（3）访问分流：技术支撑，根据分流策略向用户展示不同内容；

（4）发布运行：根据不同的实现方案进行部署；

（5）采集分析：收集数据，比较不同的方案效果，确定最终方案。

### 在abtest的应用 p值的意义，第一类和第二类错误的定义是什么

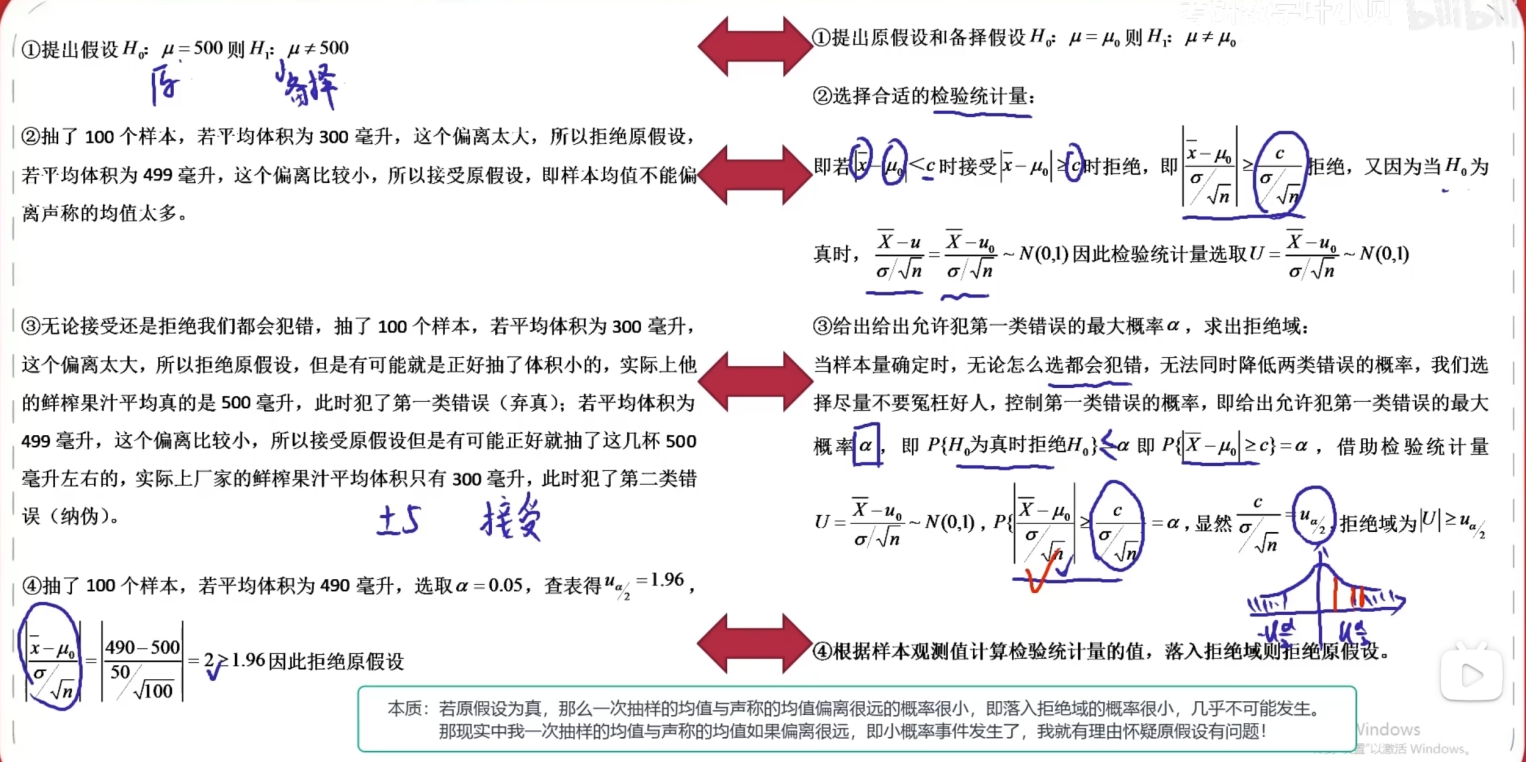
P值就是在原假设成立的前提下，出现原假设以及更极端现象的概率

**p 值 ≤ α**（通常选择 α = 0.05）：拒绝原假设，认为结果具有统计显著性。

**p 值 > α**：不能拒绝原假设，认为结果不显著。

第一类错误α叫弃真错误或显著性水平，即原假设为真时却被我们拒绝的概率；

第二类错误β叫采伪错误，即原假设为伪我们没有拒绝的概率。在一定样本量的情况下，减小一类错误必然会增大另一类错误



### 现在次日留存率是20%，执行一个策略以后，次日留存率21%，怎么证明留存率提高是因为策略影响的

### 假设检验，或者延长周期，查看七日留存率，杜绝偶然因素

要证明次日留存率的提高是由于策略的影响，可以通过假设检验来进行分析。以下是一个步骤指南，说明如何进行假设检验，以及如何延长周期查看七日留存率以减少偶然因素的影响。

1. 设定假设

原假设 (H0): 策略实施前后的次日留存率没有显著差异，即 ( p1 = p2 （20% = 21%）。

备择假设 (H1): 策略实施后次日留存率显著提高，即 p1 < p2 （20% < 21%）。

2. 收集数据

假设你有以下数据：

- 策略实施前的样本数据：

- 留存用户数：n1

- 次日留存率：20% 即0.20 \*n1

- 策略实施后的样本数据：

- 留存用户数：n2

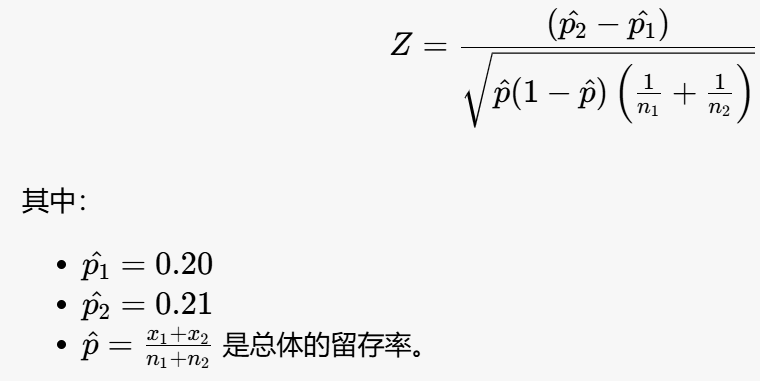
- 次日留存率：21%即0.2 \*n2

3. 选择统计检验方法

可以使用 Z 检验 来比较两个比例（留存率）。假设样本量足够大，使用 Z 检验是合适的。

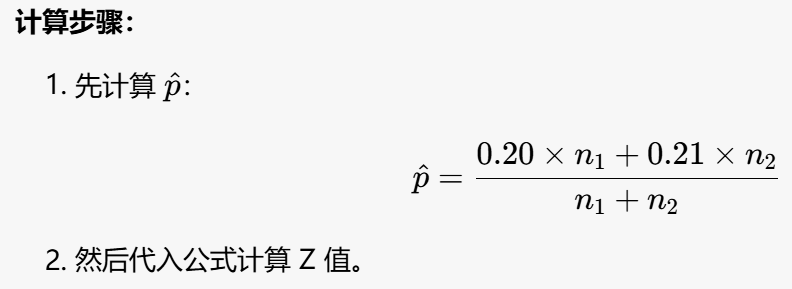
4. 计算检验统计量

使用以下公式计算 Z 值：



计算步骤：

1. 先计算 p：



5. 查找 p 值

根据计算出的 Z 值查找对应的 p 值。

6. 结果解释

- 比较 p 值与显著性水平（如 α = 0.05）：

- 如果 p 值 ≤ α，拒绝原假设，认为策略对留存率有显著影响。

- 如果 p 值 > α，不能拒绝原假设，认为留存率的变化可能是偶然因素。

7. 延长周期查看七日留存率

为了进一步确认留存率的提升不是偶然因素，可以延长观察周期，看七日留存率：

- 设定相同的假设：

- 原假设：七日留存率没有显著差异。

- 备择假设：七日留存率有显著提高。

8. 重复以上步骤

- 收集七日留存率的数据，按照相同的方式进行假设检验。

结论

通过以上步骤，可以更系统地评估策略对留存率的影响，并通过延长观察周期来减少偶然因素的影响。如果七日留存率也表现出显著提高，将进一步支持策略有效性的结论。

### 线性回归共线性、过拟合、正则化

**共线性**是指自变量之间存在高度相关性的问题。当自变量之间存在强相关性时，会导致以下问题：

不稳定性：回归系数的估计值可能会变得不稳定，导致模型对数据的微小变化敏感。

解释性降低：难以判断哪些自变量对因变量的影响更大，因为它们可能互相影响。

标准误增加：共线性会增加回归系数的标准误，使得显著性检验的结果不可靠。

解决方法

去除冗余特征：通过相关性分析，去掉相关性高的特征。

主成分回归（PCR）：使用主成分分析来降低维度，消除共线性。

岭回归：使用L2正则化来降低共线性的影响。

共线性会影响模型的稳定性和可解释性，解决方法包括去除冗余特征和使用正则化。

过拟合使模型在新数据上表现不佳，解决方法包括简化模型、交叉验证和正则化。

正则化是防止过拟合的重要技术，通过惩罚项控制模型复杂度，常见的方法有Lasso和Ridge回归。

解决过拟合和欠拟合的问题是机器学习中的重要任务。以下是一些常用的方法和策略：

解决过拟合的方法

1. 增加训练数据：

- 收集更多的数据可以帮助模型更好地学习数据的分布，从而减少过拟合。

2. 正则化：

- L1 和 L2 正则化：在损失函数中增加惩罚项，限制模型的复杂性。

- Dropout：在神经网络中随机丢弃部分神经元，防止模型对训练数据的过度依赖。

3. 简化模型：

- 选择更简单的模型或减少模型的复杂度（如减少层数或特征数）。

4. 交叉验证：

- 使用交叉验证（如 K 折交叉验证）来确保模型在不同子集上的表现一致，帮助评估模型的泛化能力。

5. 提早停止：

- 在训练过程中监控验证集的性能，当验证集性能不再提升时停止训练，防止模型在训练集上过拟合。

6. 数据增强：

- 在图像和文本等领域，通过旋转、缩放、翻转等手段生成新的训练样本，以增加数据的多样性。

解决欠拟合的方法

1. 增加模型复杂度：

- 选择更复杂的模型（如更深的神经网络或更复杂的算法），以更好地捕捉数据中的模式。

2. 减少正则化：

- 减少正则化参数的值，使模型更自由地拟合训练数据。

3. 特征工程：

- 创建新的特征或选择更有效的特征，以提高模型的表现。可以考虑使用多项式特征或其他非线性变换。

4. 增加训练时间：

- 如果模型仍未收敛，可以通过增加训练迭代次数或调整学习率来改善。

5. 使用不同算法：

- 尝试不同的机器学习算法，有时不同的算法在特定数据集上的表现会有所不同。

通过这些方法，可以有效地调整模型，以实现更好的泛化能力和预测性能。

### 5、针对实习经历问了常用的时序模型以及评价标准

## 6、对DS和DA的理解

### 数据科学（DS）

#### 定义

数据科学是一个跨学科的领域，涉及从数据中提取知识和洞察的过程。它结合了统计学、计算机科学、数学和领域知识，旨在解决复杂的问题并生成可操作的见解。

#### 主要组成部分

1. **数据收集**：从各种来源获取数据，包括结构化和非结构化数据。
2. **数据清洗与预处理**：处理缺失值、异常值和格式不一致的问题。
3. **数据分析**：使用统计和机器学习技术分析数据，识别模式和趋势。
4. **建模**：创建预测模型（如回归、分类、聚类等），并进行验证和优化。
5. **数据可视化**：通过图表和仪表盘展示分析结果，以便于理解和决策。
6. **部署与监控**：将模型部署到生产环境，并监控其性能。

### 数据分析（DA）

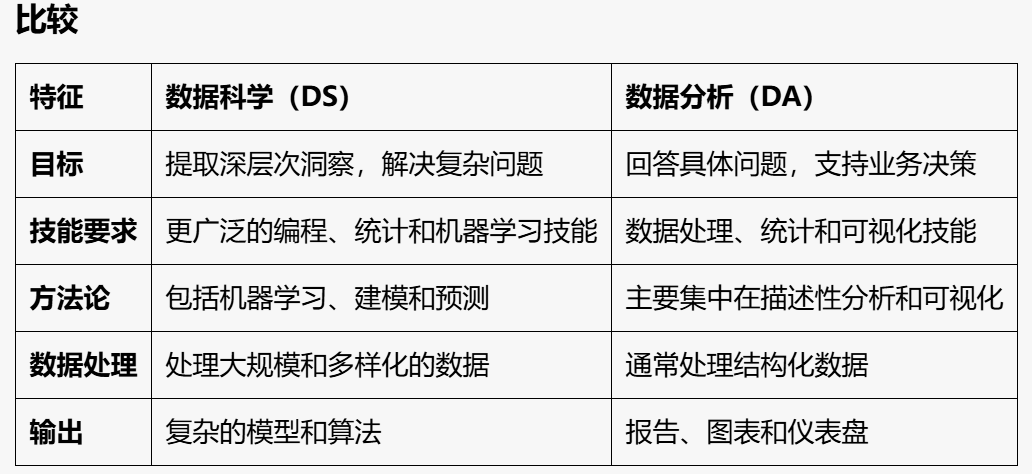
#### 定义

数据分析是数据科学的一个子集，专注于对数据进行解读和分析，以回答具体问题或支持决策。数据分析通常涉及数据的描述性统计和可视化。

#### 主要组成部分

1. **数据收集**：获取相关数据，通常是针对特定问题。
2. **数据清洗与整理**：确保数据质量，以便进行分析。
3. **描述性分析**：计算统计量（如均值、中位数、标准差）来描述数据特征。
4. **探索性数据分析（EDA）**：通过可视化和统计手段探索数据，识别潜在模式。
5. **报告生成**：将分析结果整理成报告或仪表盘，支持业务决策。

### 比较



## 数据分析常见指标

在数据分析中，常用的指标可以帮助评估和理解数据的趋势、模式和绩效。以下是一些常见的指标分类及其示例：

### 1. 描述性统计指标

* **均值（Mean）**：数据的平均值。
* **中位数（Median）**：将数据排序后，中间的值。
* **众数（Mode）**：出现频率最高的值。
* **标准差（Standard Deviation）**：数据分布的离散程度。
* **方差（Variance）**：标准差的平方，表示数据的波动性。

### 2. 计数指标

* **总计数（Count）**：数据集中总的记录数。
* **唯一计数（Distinct Count）**：数据集中唯一值的数量。

### 3. 比率和百分比

* **增长率（Growth Rate）**：某一指标的增长幅度，通常以百分比表示。
* **市场份额（Market Share）**：某一公司的销售额占市场总销售额的比例。
* **转化率（Conversion Rate）**：在特定操作中成功转化的用户比例。

### 4. 财务指标

* **净利润（Net Profit）**：收入减去所有费用后的利润。
* **毛利率（Gross Margin）**：毛利与销售额的比例。
* **投资回报率（ROI）**：投资收益与投资成本的比例。

### 5. 用户行为指标

* **用户留存率（Retention Rate）**：在特定时间段内，继续使用产品的用户比例。
* **活跃用户数（Active Users）**：在特定时间段内至少进行一次活动的用户数量。
* **客户生命周期价值（CLV）**：客户在整个生命周期内为公司带来的总价值。

### 6. 时间序列指标

* **移动平均（Moving Average）**：平滑时间序列数据的常用方法。
* **季节性指数（Seasonal Index）**：反映季节性变化对数据的影响。

### 7. 质量指标

* **缺陷率（Defect Rate）**：产品中缺陷的比例。
* **客户满意度（Customer Satisfaction Score, CSAT）**：衡量客户对产品或服务的满意程度。

### 8. 数据完整性指标

* **缺失值率（Missing Value Rate）**：数据集中缺失值的比例。
* **数据一致性（Data Consistency）**：数据在不同来源或系统中的一致性程度。

两道与实际业务结合的sql题，AB实验问题，统计学基础，挖简历

大数定律和中心极限定理、一二类错误、假设检验、p值、AB实验原理、ML分类模型评价指标等都是要掌握的基础知识  
- SQL除了刷力扣最好也刷下牛客，达到中等难度的题能不看答案做出来的水平  
- 业务理解主要体现在指标构建和思维广度与全面性方面，可以看看产品的面经补充学习下  
- 大厂真的很看实习是否垂直，这个jd听下来和上段实习的业务相似度在80％以上

字节笔试

1. 进程和线程的区别是什么?Python中何时使用多线程,何时使用多进程?

进程是系统进行资源分配和调度的基本单位，每个进程有独立的内存空间和资源。

线程是进程中的一个执行单元，多个线程共享进程的内存和资源。

**何时使用多线程**

I/O 密集型任务：如果你的程序主要受到 I/O 操作的限制（例如网络请求、文件读写），使用多线程可以有效地提高程序的响应性和处理能力，因为线程可以在等待 I/O 操作完成时切换到其他任务。

轻量级任务：当任务非常轻量且创建线程的开销不大时，使用多线程是合适的。

**何时使用多进程**

CPU 密集型任务：如果你的程序主要进行计算密集型的操作（例如数据处理、图像处理、科学计算），使用多进程可以利用多核 CPU 的优势，避免在 Python 中由于全局解释器锁（GIL）导致的性能瓶颈。

隔离性要求：当你需要确保任务之间的隔离性，避免一个任务的崩溃影响到其他任务时，使用多进程是更安全的选择。

线程进程协程、互斥锁同步锁自旋锁悲观锁乐观锁分布式锁

b.Python中list使用的是什么数据结构，其插入/删除/查询的时间复杂度分别是多少

c.如何避免模型的过拟合?

d.Word2Vec的原理?

e.常见的距离度量有那些?

2.机器学习:假设对某短视频App做了一次问卷调研，问卷中的开放题“你对App还有哪些意见或建议?"回收了约10万条用户

回答的文本，部分例子如下:

Plain Text

操作界面太复杂了，找不到观看历史在哪里

希望能增加离线缓存功能

无

好好好好好好好好

太浪费时间

请设计一套算法流程(包括流程说明和伪代码)，将这些文本自动划分成多个类别，内容相似的划分成一类，且每个类别要给出其的核心观点的总结。

# 导入必要的库

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.cluster import KMeans

from sklearn.pipeline import Pipeline

from collections import Counter

import nltk

# 假设已有用户反馈数据

user\_feedbacks = ["操作界面太复杂了，找不到观看历史在哪里希望能增加离线緩存功能",

"无",

"好好好好好好好好",

"太浪费时间"]

# 数据预处理

def preprocess\_text(feedback):

# 清洗文本

feedback = feedback.replace("无", "").strip()

# 分词、去除停用词

tokens = nltk.word\_tokenize(feedback)

tokens = [word for word in tokens if word not in stop\_words] # 停用词列表

return ' '.join(tokens)

cleaned\_feedbacks = [preprocess\_text(feedback) for feedback in user\_feedbacks]

# 特征提取

vectorizer = TfidfVectorizer()

X = vectorizer.fit\_transform(cleaned\_feedbacks)

# 聚类

num\_clusters = 3 # 假设我们设定要分为5个类别

kmeans = KMeans(n\_clusters=num\_clusters)

kmeans.fit(X)

# 获取聚类结果

clusters = kmeans.labels\_

# 核心观点提取

def extract\_core\_views(clustered\_feedbacks, clusters):

core\_views = {}

for cluster in set(clusters):

cluster\_feedbacks = [feedback for feedback, cluster\_id in zip(clustered\_feedbacks, clusters) if cluster\_id == cluster]

word\_counts = Counter(" ".join(cluster\_feedbacks).split())

core\_views[cluster] = word\_counts.most\_common(3) # 获取前三个高频词

return core\_views

core\_views\_summary = extract\_core\_views(cleaned\_feedbacks, clusters)

# 输出结果

for cluster, core\_views in core\_views\_summary.items():

print(f"类别 {cluster}: {core\_views}")

广告、电商、saas