一文搞定机器学习算法

第1部分 机器学习中的统计学

第0章 统计学

0.1 常见分布

0.1.1 正态分布

1、数字特征
若 $X \sim N(\mu, \sigma^2)$，则
期望：$E(X) = \mu$；
方差：$Var(X) = \sigma^2$。
注：标准正态分布记为 $X \sim N(0, 1)$

2、性质
① 每一个特定正态分布均可通过其均值μ、标准差σ来区分。他们分别确定了分布的位置和形状。
② 正态曲线的最高点在均值处，均值也是分布的中位数和众数。
③ 不同分布的均值可以是任意数值：负数、零或正数。
④ 正态概率分布是对称的。均值左边的线形是均值右边的线形的镜像。曲线的尾端向两个方向无限延伸，且理论上永远不会与横轴相交。
⑤ 标准差决定曲线的宽度。标准差值大产生较宽、较平的曲线，表明数据有更大的变异。
⑥ 正态概率分布曲线下的总面积是1，对所有的连续型概率分布都是如此。
⑦ 正态随机变量的概率由曲线下面积给出。

0.1.2 t分布

1、什么时候服从t分布？
当样本量较小，标准差未知，但已知来自正态总体时，样本服从t分布。

2、t分布的性质

① 分布左右对称；

② 最高点矮于N(0,1)，尾巴厚于N(0,1)；

③ 期望为0，方差 $\frac{n}{n-2}$ ；

④ 自由度越大，曲线分散程度越小，即越高窄；

⑤ 正态分布是与自由度无关的一条曲线；t分布是依自由度而变的一组曲线。

0.1.3 两点分布

1、什么是伯努利试验？

① 在同样的条件下，重复的相互独立的进行的一种随机试验。

② 随机试验的结果只有两种结果（成功，或者失败）。

2、数字特征

若满足$X \sim B(1,p)$，则有

期望：E(X) = p；

方差：Var(X) = p(1-p) = pq；

其中，p为每一次事件成功的概率，q为每一次事件失败的概率。

0.1.4 二项分布

1、什么是二项分布？

① 做某件事的次数是固定的，且n次事件是相互独立的（次数用n表示）。

② 每一次事件都有两个可能的结果（成功，或者失败）。

③ 每一次事件成功的概率都相等，成功的概率用p表示。

④ 求发生k次的概率，服从二项分布。

注: 伯努利分布是n=1时的二项分布的特殊情况。

2、数字特征

若满足 $X \sim B(n,p)$，则有

期望：E(X) = np ；

方差：Var(X) = np(1-p) = npq；

其中，p为每一次事件成功的概率，q为每一次事件失败的概率。

0.1.5 χ^2分布

0.1.6 泊松分布

1、什么是泊松分布？
① 发生的事件是独立事件。
② 在任何相同的时间范围内，某事件发生的概率相同。
③ 求某个时间范围内，发生某事件k次的概率，服从泊松分布。
注: 当二项分布的n很大而p很小时，泊松分布可作为二项分布的近似，其中λ为np。通常当n≧20,p≦0.05时，就可以用泊松公式近似得计算。事实上，泊松分布正是由二项分布推导而来的。

2、数字特征
若满足 $X \sim P(\lambda)$，则有
期望：λ；
方差：λ；
其中，参数λ是单位时间(或单位面积)内随机事件的平均发生率。
泊松分布适合于描述单位时间内随机事件发生的次数。

0.1.7 F分布

0.2 假设检验

0.2.1 假设检验的原理

小概率事件在一次实验中几乎是不可能发生的。
如果一个小概率实验在一次随机试验中发生了，那么我们有理由拒绝原假设！

0.2.2 假设检验的流程

① 根据实际问题的要求，提出原假设和备择假设；
② 给定显著性水平 α 以及样本容量n；
③ 确定检验统计量及拒绝域的形式；
④ 按P{当 H_0 为真拒绝 H_0} ≤ α 取样；
⑤ 根据样本观察值做出决策，是接受原假设还是拒绝原假设。

0.2.3 t检验

0、使用场景
t检验的前提是要求样本服从正态分布或近似正态分布，不然可以利用一些变换（取对数、开根号、倒数等等）试图将其转化为服从正态分布是数据，如若还是不满足正态分布，只能利用非参数检验方法。
当样本量大于30的时候，可以认为数据近似正态分布。

1、单样本均值检验（One-sample t-test）
用于检验 总体方差未知、正态数据或近似正态的 单样本的均值 是否与 已知的总体均值相等。

2、两独立样本均值检验（Independent two-sample t-test）
用于检验 两对独立的 正态数据或近似正态的 样本的均值 是否相等，这里可根据总体方差是否相等分类讨论。

3、配对样本均值检验（Dependent t-test for paired samples）
用于检验 一对配对样本的均值的差 是否等于某一个值。

4、回归系数的显著性检验（t-test for regression coefficient significance）
用于检验 回归模型的解释变量对被解释变量是否有显著影响。

0.2.4 样本量的确定

$$n = \frac{(Z_\alpha + Z_\beta)^2 * 2\sigma^2}{\delta^2}$$

n为样本量，Z_α 双侧临界值，α 一般为0.05，Z_β为单侧临界值，σ 为标准差，δ 为实验组与对照组的差值。

0.2.5 显著性水平α与p值

0.2.5.1 显著性水平α的含义

在假设检验中，它的含义是当原假设正确时却被拒绝的概率或风险，其实这就是假设检验中犯弃真错误的概率。

0.2.5.2 α的选择

一般选择0.1，0.05，0.01

0.2.5.3 p值

假设检验问题的p值是由检验统计量的样本观测值得出的原假设可被拒绝的最小显著性水平;
P值是当原假设为真时，样本观察结果或更极端结果出现的概率。
P值越小，拒绝原假设的理由越充分。

0.2.5.4 p值的决定因素

①样本数据与原假设之间的差异；
②样本量；
③被假设参数的总体分布。

0.2.5.5 显著性水平α与p值

确定α以后，拒绝域的位置也就相应确定了，其好处是进行决策的界限清晰，但缺陷是进行决策面临的风险是笼统的。
p值的长处是它反映了观察到的实际数据与原假设之间不一致的概率值，与传统的拒绝域范围相比，p是一个具体的值，这样就提供了更多的信息。

对于任意给定的显著性水平 α:
① 若p值≤ α，则在显著性水平 α 下拒绝原假设；
② 若p值＞ α，则在显著性水平 α 下接受原假设。

p值表示反对原假设的依据的强度，p值越小，反对原假设的依据越强、越充分。
一般若p值≤0.01，称推断拒绝原假设的依据很强或称检验是高度显著的；
若0.01≤p值≤0.05，称推断拒绝原假设的依据是强的或称检验是显著的；
若0.05≤p值≤0.1，称推断拒绝原假设的理由是弱的或称检验是不显著的；
若p值＞0.1，一般来说没有理由拒绝原假设。

0.2.6显著性检验

只对犯第Ⅰ类错误的概率加以控制，而不考虑犯第二类错误的概率的检验称为显著性检验。

0.2.7 两类错误

0.2.7.1 两类错误的定义

第Ⅰ类错误是原假设H_0为真却被我们拒绝了，犯这种错误的概率用α表示，所以也称α错误或弃真错误；
第Ⅱ类错误是原假设为伪我们却没有拒绝，犯这种错误的概率用β表示，所以也称β错误或取伪错误。

0.2.7.2 两类错误的概率

对于一定的样本量n，如果减少α错误，就会增大犯β错误的机会；若减少β错误就会增大犯α错误的机会。
增大样本量可以同时减少α和β(贾俊平说)。(腾讯面试官说减少的是β)

0.2.7.3 两类错误的控制

一般来说，哪一类错误所带来的后果越严重，危害越大，在假设检验中就应当把哪一类错误作为首要的控制目标。
在假设检验中，一般首先控制犯α错误。主要的原因在于，原假设是什么通常是明确的，而备择假设是什么通常是模糊的。

0.2.7.4 两类错误的地位

在进行显著性检验时，犯第Ⅰ类错误的概率是由我们控制的，α 取得小，则概率P{当 H_0 为真拒绝 H_0}就越小，这保证了当原假设为真时错误地拒绝原假设的可能性很小。这意味着原假设是受到保护的，也表明原假设和备择假设的地位不是对等的。

0.2.8 "不拒绝" 与 "接受"

从假设检验的原理看，不拒绝原假设意味着我们所构造的与原假设相矛盾的事件没有发生，但可能会有许多其他的与原假设矛盾的小概率事件，我们没有也无法证实所有的这些小概率事件不会发生，因此，我们把假设检验中出现接受 H_0 的结果解释为“没有发现充足的证据反对 H_0”，或更严格地的解释为，“在显著性水平 α 下没有发现充足的证据反对 H_0”，而不是“接受原假设 H_0”，因为我们无法证明原假设是正确的。即接受备择假设一定意味着原假设错误；没有拒绝原假设并不能表明备择假设一定是错的

0.3 回归分析

0.3.1 gauss-markov 假定

①$E(\varepsilon)=0$;
②$var(\varepsilon_i)=\sigma, i=1,2,\cdots$;
③$\varepsilon_i \sim N(0,\sigma), i=1,2,\cdots$;
④$cov(\varepsilon_i,\varepsilon_j)=0, i\neq j, i,j=1,2,\cdots$

0.3.2 方程的显著性检验

原假设与备择假设
检验统计量（F统计量）

0.3.3 系数的显著性检验

原假设与备择假设
检验统计量（t统计量）

0.3.4 多重共线性的判别和处理

1、判别：
①模型中各对自变量之间显著相关；
②当模型的线性关系检验(F检验)显著时，几乎所有的回归系数的t检验却不显著；
③回归系数的正负号与预期的相反；
④容忍度与方差扩大因子。容忍度越小，多重共线性越严重；方差扩大因子等于容忍度的倒数，方差扩大因子越大，多重共线性越严重，一般认为，方差扩大因子大于10时，存在严重的多重共线性。

2、处理：
①将相关的自变量从模型中剔除，使保留的自变量尽可能不相关；
②如果要在模型中保留所有自变量，那就应该:
ⅰ 避免根据t统计量对单个参数进行检验；
ⅱ 对因变量y的推断(估计或预测)限定在自变量样本值的范围内。

0.4 时间序列

0.4.1 时间序列的组成要素

①长期趋势：时间序列在长期内呈现出来的某种持续上升或持续下降的变动，也称长期趋势；
②周期性：时间序列中呈现出来的围绕长期趋势的一种波浪形或震荡式变动。循环波动无固定规律，变动周期多在一年以上，且周期长短不一；
③季节性：时间序列在一年内重复出现的周期性波动；
④随机性：时间序列中除去趋势、周期性和季节性之后的偶然性波动。

0.4.2 常用的预测模型

1、类型：加法模型、乘法模型
2、模型：AR(p)，MA(q)，ARMA(p,q)，ARIMA(p,d,q)，prophet

0.5 损失函数

常用的损失函数

0.6 评估指标

0.6.1 分类问题评估指标

1、分类问题中常用的评价指标？
ROC-AUC，Macro-F1，Micro-F1，Precision，Recall

2、ROC-AUC计算原理？

3、Macro-F1，Micro-F1的区别？

4、多标签学习中的评估指标？
Ranking Loss，Coverage，Hamming Loss，Average Precision， One Error

0.6.2 回归问题评估指标

1、回归问题中常用的评价指标?

均方损失（RSS），绝对值损失（MAE），R^2

0.6.3 聚类问题评估指标

聚类问题中的评价指标？
sihouette，SSE，CH-index

第2部分 传统机器学习

第一章 树模型

1.1 基本知识

1.1.1 熵

在物理学中表示一个系统的混乱程度，在信息论里面表示随机变量的不确定性的度量，熵越大，不确定性越大。

$$H(p) = -\sum_{i=1}^{n} p_i log(p_i)$$

显然$0 \leq H(p) \leq log(n)$，当且仅当$p_1 = p_2 = \cdots = p_n$时，$H(p) = log(n)$。

1.1.1.1 条件熵

在给定随机变量X的情况下，随机变量Y的不确定性

$$H(Y|X) = \sum_{i=1}^{n} P(X = x_i) H(Y|X = x_i), i = 1, 2, \cdots, n$$

1.1.1.2 联合熵

它是联合概率分布$P(X, Y)$的经验熵

$$H(X,Y) = -\sum_{i,j} P(X=x_i, Y=y_j) log(P(X=x_i, Y=y_j))$$

1.1.1.3 交叉熵

记数据集的真实分布为 p，估计分布为 q，若用 q来拟合 p，则交叉熵损失为

$$H(p,q) = -\sum_{i=1}^{n} p_i log(q_i)$$

1.1.1.4 KL散度

用于衡量两个分布 p, q 之间的差异程度

$$\begin{aligned} D_{KL}(p|q) &= \sum_{i=1}^{n} p_i log(\frac{p_i}{q_i}) \\ &= \sum_{i=1}^{n} p_i log(p_i) - \sum_{i=1}^{n} p_i log(q_i) \\ &= -H(p) + H(p,q) \end{aligned}$$

显然 KL 散度是交叉熵和经验熵的差值。

1.1.2 信息增益

特征 A 对训练集 D 的的信息增益 $g(D,A)$ 定义为集合 D 的经验熵 $H(D)$与集合 D 在给定特征 A条件下的条件熵 $H(D|A)$之差

$$g(D,A) = H(D) - H(D|A)$$

一般的，熵 $H(Y)$ 与条件熵 $H(Y|X)$之差称为互信息，在决策树里，信息增益等价于训练数据集中类别和特征的互信息。

1.1.3 信息增益率

特征 A 对训练数据集 D 的信息增益率 $g_R(D,A)$ 定义为其信息增益 $g(D,A)$ 与训练数据集

D 关于特征 A 的值的熵 $H_A(D)$ 之比，即

$$g_R(D,A)=\frac{g(D,A)}{H_A(D)}$$

其中

$$H_A(D)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}log_2\frac{|D_i|}{|D|}$$

n 为特征 A 的取值个数。

1.1.4 Gini 指数

对于一个 K 分类问题，样本的概率分布为 p_k，则该分布的基尼指数为

$$\begin{aligned}Gini(p)&=\sum_{i=1}^{n}p_k(1-p_k)\\&=1-\sum_{i=1}^{n}p_k^2\end{aligned}$$

基尼指数 $Gini(D)$ 代表集合 D 的不确定性，基尼指数$Gini(D,A)$ 表示经 $A=a$ 分割后集合 D 的不确定性，基尼指数之越大，样本集合的不确定性也越大。

1.2 决策树

决策树是一个树状结构算法，它是一套 if else 规则，常用于**解决分类问题**。对于一个新样本，在经过层层 if else 规则之后，最终落在哪个叶子节点上，它就属于该节点所在的类别。决策树的由**特征选择、树的生成、树的剪枝**三步组成。常见的决策树算法有 **ID3**，**C4.5**，**CART**。

1.2.1 ID3 算法

$ID3$ 算法的核心是在决策树各个节点上用**信息增益准则**选择特征，递归地构建决策树。具体

地，从根结点开始，对结点计算所有可能的特征的信息增益，选择信息增益最大的特征作为节点的特征，由该结点的不同取值建立子节点，再对子结点递归调用上述方法，构建决策树，直到所有特征的信息增益均很小，或没有特征可以选择为止，最后得到决策树。$ID3$**相当于是用极大似然法进行概率模型的选择**。

注意：
(1) $ID3$只能处理**离散特征**;
(2) 特征在不同层级之间**不能反复使用**;
(3) 使用信息增益来做特征选择，容易导致**偏向于选择属性值多的特征**，从直观上也能理解，一个特征属性值越多，包含的信息也就越多，用它来做分裂结点，数据集的不确定性降低地更快。

1.2.2 C4.5 算法

相较于 $ID3$ 算法，$C4.5$ 只是在选择分裂的时候用**信息增益率**来做特征选择。

注意：
(1) 相较于 $ID3$，$C4.5$ 既能处理离散特征**也能处理连续特征**;
(2) 使用信息增益率做特征选择，避免了偏向选择属性值较多的特征的问题;
(3) $C4.5$ 在处理连续型特征时，通过对数据排序之后找到类别不同的分割线作为切分点，根据切分点把连续型特征转换为多个取值区间上的离散型特征;
(4) $C4.5$ 一般来说也是**多叉树**。

1.2.3 CART算法

$CART$ 全称是分类与回归树(Classify and Regression Tree)，它既可以用于分类也可以用于回归，$CART$ 算法中假设树是**二叉树**，内部节点特征取值为 “是” 和 “否”，左边是取值为 “是” 的分支，右边是取值为 “否” 的分支。

$CART$ 算法由两步组成：
(1) 决策树的生成：基于训练数据生成决策树，生成的决策树要尽量**大**;
(2) 决策树的剪枝：用验证数据集对已生成的决策树进行剪枝并选择最优子树，用**损失函数最小**作为选择标准。

注意：
(1) $CART$ 既可以做分类也可以做**回归**;
(2)不同于 $ID3$、$C4.5$ 可以是多叉树，$CART$ 只能是**二叉树**;
(3) $ID3$ 和 $C4.5$ 的特征是不能在不同层级之间复用的，而 $CART$ 可以;

(4) $C4.5$ 通过剪枝来平衡模型的准确性和泛化能力，而 $CART$ 直接**利用全部数据**发现所有可能的树进行对比.

1.2.4 决策树的剪枝

决策树生成算法递归地生成决策树，直到满足停止条件。这种情况下，对于训练集拟合效果较好，但在测试集上的效果往往不理想，即容易产生过拟。因此要对决策树进行剪枝，简化树状结构，提高泛化能力。

剪枝分为预剪枝和后剪枝。
预剪枝：在树的构建生长阶段进行剪枝，如果结点满足信息 $Gini$ 指数小于阈值或者该结点的样本数量少于特定值，则不再对该结点继续细分。
后剪枝：在树完全生长后再从叶子结点逐个往上剪枝，利用测试集数据验证剪枝前后，树的损失函数是否有变化。如果剪枝后损失函数减小，说明剪枝提高了模型泛化能力，否则不能剪枝。

1.3 面试题

1.3.1 简述决策树原理

决策树是一类常见的机器学习方法，它是基于树的结构进行决策的。每次做决策时选择最优划分属性，一般而言，随着划分过程不断进行，我们希望决策树的分支节点所包含的样本尽可能属于同一个类别，即节点的“纯度”(purity)越来越高。

决策树学习算法包含**特征选择、决策树的生成与剪枝过程**。决策树的学习算法通常是递归地选择最优特征，并用最优特征对数据集进行分割。开始时，选择最优特征构建根节点，该特征有几种取值就分割为几个子集，每个子集分别递归调用此方法，返回节点，返回的节点就是上一层的子节点。直到达到终止条件（预剪枝条件等）。

关于熵和信息增益等的介绍见第1节。

常用的决策树算法有 $ID3$，$C4.5$ 和 $CART$，其中 $ID3$ 以信息增益为准则来选择划分属性，$C4.5$ 使用信息增益率来进行属性的选择，$CART$ 使用基尼系数来选择划分属性。

1.3.2 简述决策树的构建过程

数据预处理，根据信息增益或信息增益率准则选择分裂结点，树的剪枝。

1.3.3 简述信息增益率的优缺点

优点：$ID3$ 使用信息增益准则对决策树进行划分，其偏向于选择分支较多的特征分裂，因为具有较多属性值的特征分裂后其信息增益通常较大。此时就会有较大的概率出现每个分支节点只包含一个样本的情况，因为分支节点的纯度达到最大，但显然这种决策树的泛化能力较差，对新样本的预测能力较弱。信息增益率是对信息增益的一种改进，对属性值多的样本做一下惩罚，避免分类的时候偏向于优先划分属性值多的特征。

缺点：信息增益率偏向取值较少的特征，当特征取值较少时，信息增益率的计算公式中分母的值较小，因而信息增益率比较大。所以它偏向取值较少的特征。

我们在使用信息增益率时，通常也并不是直接使用。而是先在候选特征中找出信息增益高于平均水平的特征，然后在这些特征中再选择信息增益率最高的特征。

1.3.4 $C4.5$ 对 $ID3$ 做了哪些改进

ID3 算法是采用**信息增益**作为评价标准进行分支的决策树算法。

ID3的缺点：
(1) 它对取值较多的属性非常敏感，例如，数据集中的某个属性取值是不重复的，如果我们用这个属性来划分数据集，那么它的信息增益会很大，但这并不是我们想要的结果，因为它的泛化能力很差。
(2) ID3算法不能处理连续型属性。
(3) ID3算法不能处理属性具有缺失值的样本。
(4) ID3算法会生成很深的树，容易过拟合。

C4.5算法主要对ID3作出了以下方面的改进：
(1) 用信息增益率来选择属性，克服了用信息增益来选择属性时偏向选择值多的属性的不足。
(2) 可以处理连续数值型属性。
(3) 使用PEP(Pessimistic Error Pruning)的后剪枝策略。
(4) 能够处理缺失值。

处理的方式通常有三种：

① 赋上该属性最常见的值或者取均值；

② 丢弃有缺失值的样本；

③ 根据节点的样例上该属性值出现的情况赋一个概率，比如该节点上有10个样本，其中属性A的取值有6个为“高”，4个为“低”。那么对该节点上缺失的属性A，以0.6的概率设为“高”，0.4的概率设为“低”。

C4.5的缺点：
(1) 算法低效，在构造树的过程中，需要对数据集进行多次的顺序扫描和排序（主要在于对连续特征的处理上）。
(2) 内存受限，适合于能够驻留于内存的数据集，当训练集大到无法在内存中容纳时，程序无法运行。
(3) 无论是 ID3 还是 C4.5 最好在小数据集上使用，决策树分类一般只适用于小数据集。当属性取值很多时最好选择 C4.5 算法，而ID3 的效果会非常差。

1.3.5 $C4.5$ 是如何处理连续数值型属性的？

C4.5 既可以处理离散型特征，也可以处理连续型特征。对于连续分布的特征，其处理方法是：先把连续特征转换为离散特征再进行处理。虽然本质上特征的取值是连续的，但对于有限的采样数据它是离散的，如果有N个样本，那么我们就有N-1种离散化的方法(**连续属性的离散化通常是分成两类**，即将小于等于某一个分割点的样本分为一类，大于某一个分割点的样本分为一类)。具体分割点的选择依赖于信息增益，选择具有最大信息增益的分割点。另外，在分割之前，需要对连续属性进行排序（升序），这样可以大大减少运算量。经证明，在**决定连续特征的分界点时**采用**信息增益**这个指标，而**选择属性**的时候才使用**信息增益率**这个指标能选择出最佳分类特征。

具体步骤如下：
step1: 对特征的取值进行升序排序。
step2: 以两个取值之间的中点作为可能的分裂点，将数据集分成两部分，计算每个可能的分裂点的信息增益。优化算法就是只计算分类属性发生改变(对于升序排序的特征取值而言，对每一个分裂点都进行计算未免太过耗时，实践证明，取那些夹在相同分类属性之间的分裂点必不会时全局最小)的那些特征取值。
step3: 选择修正后信息增益最大的分裂点作为该特征的最佳分裂点。
step4: 计算最佳分裂点的信息增益率作为特征的信息增益率。

分裂点的信息增益修正方法：
将信息增益减去 $\frac{log_2(N-1)}{|D|}$ 得到修正后的信息增益，其中N是连续特征的取值个数，D是训练数据数目。此修正的原因是：当离散特征与连续特征并存时，C4.5算法倾向于选择连续特征做最佳树分裂点。

1.3.6 $C4.5$ 与 $CART$ 的区别

(1) $C4.5$ 只能做分类，$CART$ 既可以做分类也可以做回归，分类的时候用 $Gini$ 指数，回归的时候用平方差。

(2) $C4.5$ 可以是多叉树，而 $CART$ 只能是二叉树。
(3) $C4.5$ 的特征在每个层级之间不会复用，$CART$ 的每个特征可以复用。
(4) $C4.5$ 通过剪枝来平衡模型的准确性和泛化能力，而 $CART$ 直接利用全部数据发现所有可能的树进行对比。

1.3.7为什么要对决策树进行剪枝？

避免决策树过拟合(Overfitting)样本。$ID3$、$C4.5$、$CART$ 算法生成的决策树非常详细并且庞大，每个属性都被详细地加以考虑，决策树的树叶节点所覆盖的训练样本都是“纯”的。如果用这样的决策树对训练样本进行分类，其在训练样本上的表现非常好，误差率极低。但由于训练集中的错误数据也会被决策树学习，成为决策树的一部分，因而在测试集上的表现就没有那么好，甚至会很差，这就是所谓的过拟合(Overfitting)问题。Quinlan教授试验，在数据集中，过拟合的决策树的错误率比经过剪枝的决策树的错误率要高。

1.3.8 决策树的剪枝策略？

预剪枝

树的构造将提前停止，如果
(1) 树到达一定高度；
(2) 节点下包含的样本点数量小于一定数目；
(3) 信息增益小于一定的阈值等等；
(4) 节点下所有样本都属于同一个类别。

后剪枝

后剪枝的剪枝过程是删除一些子树，然后用其叶子节点代替，这个叶子节点所标识的类别通过大多数原则(majority class criterion)确定。所谓大多数原则，是指剪枝过程中，将一些子树删除而用叶节点代替，这个叶节点所标识的类别用这棵子树中大多数训练样本所属的类别来标识，所标识的类称为majority class。

后剪枝首先通过完全分裂构造完整的决策树，允许过拟合，然后采取一定的策略来进行剪枝，常用的后剪枝策略包括：
(1) 降低错误剪枝 REP(Reduced Error Pruning);
(2) 悲观错误剪枝 PEP(Pessimistic Error Pruning);
(3) 基于错误剪枝 EBP(Error Based Pruning);
(4) 代价-复杂度剪枝 CCP(Cost Complexity Pruning);

(5) 最小错误剪枝 MEP(Minimum Error Pruning)

剪枝的准则：
(1) 使用训练集 (Training Set) 和验证集 (Validation Set) 来评估剪枝方法在修剪结点上的效用;
(2) 使用所有的训练集进行训练，但是用统计测试来估计修剪特定结点是否会改善训练集合外的数据的评估性能，如使用Chi-Square(Quinlan, 1986)测试来进一步扩展结点是否能改善整个分类数据的性能，还是仅仅改善了当前训练集合数据上的性能。
(3) 使用明确的标准来衡量训练样例和决策树的复杂度，当编码长度最小时，停止树增长，如MDL(Minimum Description Length)准则。

Reduced-Error Pruning(REP,错误率降低剪枝)

自底向顶剪枝：该剪枝方法考虑将树上的每个节点作为修剪的候选对象，决定是否修剪这个结点由如下步骤组成
step1：删除以此结点为根的子树;
step2：使其成为叶子结点;
step3：赋予该结点关联的训练数据的最常见分类;
step4：当修剪后的树对于验证集合的性能不会比原来的树差时，才真正删除该结点。

因为训练集合的过拟合，使得验证集数据能够对其进行修正，反复进行上面的操作，从底向上的处理结点，删除那些能够最大限度的提高验证集的精度的结点，直到进一步修剪有害为止(有害是指修剪会减低验证集的精度)

REP是最简单的后剪枝方法之一，不过在数据量比较少的情况下，REP方法趋于过拟合而较少使用。这是因为训练集中的特性在剪枝过程中被忽略，所以在**验证数据集比训练数据集合小的多时**，要注意这个问题。

尽管REP有这个缺点，不过REP仍然作为一种基准来评价其它剪枝算法的性能。它对于两阶段决策树学习方法的优点和缺点提供了了一个很好的学习思路。由于验证集合没有参与决策树的创建，所以用REP剪枝后的决策树对于测试样例的偏差要好很多，能够在一定程度上解决过拟合问题。

Pessimistic Error Pruning(PEP, 悲观错误剪枝)

该算法被认为是当前决策树后剪枝算法中精度比较高的算法之一，但是仍存在有缺陷。首先，PEP算法是唯一使用Top-Down剪枝策略，这种策略会导致与预剪枝出现同样的问题，即将该结点的某子节点不需要被剪枝时被剪掉；另外PEP方法会有剪枝失败的情况出现。

虽然PEP方法存在一些局限性，但是在实际应用中表现出了较高的精度。另外，PEP方法不需

要分离训练集和验证集，对于数据量比较少的情况比较有利。再者，其剪枝策略相比于其它方法效率更高，速度更快。因为在剪枝过程中，树中的每棵子树最多需要访问一次，在最坏情况下，它的计算时间复杂度也只和非剪枝树的非叶子节点数目成线性关系。

1.3.9 简述分类树与回归树的联系与区别

分类树：以C4.5分类树为例，C4.5分类树在每次分枝时，是穷举每一个feature的每一个阈值，找到使得按照feature<=阈值，和feature>阈值分成的两个分枝的集合信息熵最小的阈值，按照该标准分枝得到两个新节点，用同样方法继续分枝直到得到类别唯一的叶子节点，或达到预设的终止条件，若最终叶子节点中的类别不唯一，则以占有最多数的类别作为该叶子节点的最终分类类别。

回归树：回归树总体流程与分类树类似，区别在于，回归树的每个节点（不一定是叶子节点）都会得一个预测值，以年龄为例，该预测值等于属于这个节点的所有人年龄的平均值。分枝时穷举每一个feature的每个阈值找最好的分割点，但衡量最好的标准不再是信息熵(信息增益)，而是最小化均方误差即$\frac{1}{N}\sum_{i=1}^{N}(第i个人的年龄-预测年龄)^2$。也就是被预测出错的人数越多，错的越离谱，均方误差就越大，通过最小化均方误差能够找到最可靠的分枝依据。分枝直到每个叶子节点上人的年龄都唯一或者达到预设的终止条件(如叶子个数上限)，若最终叶子节点上人的年龄不唯一，则以该节点上所有人的平均年龄做为该叶子节点的预测年龄。

总结：
(1) 分类树使用**信息增益或信息增益率**来划分节点；每个节点样本的类别情况**投票**决定测试样本的类别。
(2) 回归树使用**最小均方误差**划分节点；每个节点样本的**均值**作为测试样本的回归预测值。

1.3.10 $CART$ 如何生成回归树？

1.3.11 $CART$ 对取值数目≥ 3 的离散特征如何处理？

因为 $CART$ 是二叉树，所以对于样本中有 $N \geq 3$ 个取值的离散特征处理时也只能有两个分支，这就要通过人为的创建二取值序列并取 $GiniGain$ 最小者作为树分叉决策点。如某特征值具有 ['young', 'middle', 'old'] 三个取值，那么二分序列会有如下3种可能性(空集和满集在 $CART$ 分类中没有意义):

[(('young'), ('middle', 'old')), (('middle'), ('young', 'old')), (('old'), ('young', 'middle'))]

采用 $CART$ 算法，就需要分别计算按照上述 List 中的二分序列做分叉时的 $Gini$指数，然后选取产生最小的 $GiniGain$ 的二分序列做该特征的分叉二值序列参与树构建的递归。

CART不适用于离散特征有过多取值的场景。若一定要使用CART，最好预先人为的将离散特征的取值缩减。 那么对于二分后的左右分支，如果特征取值tuple中元素多于2个，该特征一般还是需要继续参与到子数据集的分割中去。这在一定程度上有助于提升模型的精度(一定意义上改进了C4.5完全的贪心算法，使得模型有一定的全局（局部）特征考量的性质)。

1.3.12 决策树如何处理缺失值？

缺失值问题可以从四个方面来考虑

(1)在进入模型开始训练之前，我们可以对缺失值做上一定的处理。
① 为缺失值赋上该属性最常见的值或者取均值；
② 如果数据集较大，可以丢弃有缺失值的样本；
③ 根据节点的样例上该属性值出现的情况赋一个概率，比如该节点上有10个样本，其中属性A的取值有6个为“高”，4个为“低”。那么对该节点上缺失的属性A，以0.6的概率设为“高”，0.4的概率设为“低”。

(2) 训练样本在分裂属性上存在缺失值如何处理？(计算分裂损失减少值时，忽略特征缺失的样本，最终计算的值乘以比例(实际参与计算的样本数除以总的样本数))。

假如使用ID3算法，那么选择分类特征时，就要计算所有特征的熵减(信息增益，Gain)。假设10个样本，特征是A，B，C。在计算 A 特征的熵时发现，第10个样本的 A 特征取值缺失，那么就把第10个样本去掉，用前9个样本组成新的样本集，在新样本集上按正常方法计算 A 特征的熵减。然后结果乘0.9（未缺失样本的比例 $\frac{9}{10}$），就是 A 特征分裂最终的熵。

(3) 分类特征选择完成，对训练样本分类，发现样本特征缺失怎么办？（将该样本分配到所有子节点中，权重由1变为具有特征 A 的样本划分成的子集样本个数的相对比率，计算错误率的时候，需要考虑到样本权重)。

① 单独为属性缺失的样本划分一个分支子集。
② 比如该节点是根据 A 特征划分，但是待分类样本 A 特征缺失，怎么办呢？假设 A 特征离散，有1、2两种取值，那么就把该样本分配到两个子节点中去，但是权重由1变为相应离散值个数占样本的比例，然后计算错误率的时候乘以样本权重。注意，不是每个样本都是权重为1，存在分数。

(4) 训练完成，给测试集样本分类，有缺失值怎么办？(分类时，如果待分类样本有缺失变量，而决策树决策过程中没有用到这些变量，则决策过程和没有缺失的数据一样；否则，如果决策要用

到缺失变量，决策树也可以在当前节点做多数投票来决定(选择样本数最多的特征值方向))

① 如果有单独的缺失分支，使用此分支。
② 给待分类样本的 A 特征分配一个最常出现的属性值，然后进行分支预测。
③ 根据其他特征为该待分类样本填充一个特征a值，然后进行分支处理。
④在决策树中 A 特征节点的分支上，遍历 A 特征节点的所有分支，探索可能所有的分类结果，然后把这些分类结果结合起来一起考虑，按照概率决定一个分类。
⑤ 待分类样本在到达 A 特征节点时就终止分类，然后根据此时 A 节点所覆盖的叶子节点类别状况为其分配一个发生概率最高的类。

1.3.13 如果决策树属性用完了仍未对决策树完成划分应该怎么办？

当训练集很大以及特征数量并不是很多的情况下，这种状况很容易发生。此时的解决方法一般有两种：
(1) 一种即退出模型，做**特征组合、交叉**等操作扩充特征。当有了足够多的特征后，便可以有效改善这种情况。但在一般情况下，树模型如果特征过多，相当容易过拟合。
(2) 另一种方法便是直接对叶子节点采用**“多数表决”**，即使用节点中出现最多的类别作为该节点的标签。

1.3.14 决策树中过拟合出现的原因以及如何避免过拟合？

原因：
(1) 在决策树构建的过程中，对决策树的生长没有进行合理的限制(剪枝)；
(2) 在建模过程中使用了**较多的输出变量**，变量较多也容易产生过拟合；
(3) 样本中有一些**噪声数据**，噪声数据对决策树的构建的干扰很多，没有对噪声数据进行有效的剔除。

解决方法:
(1) 选择合理的参数(树的深度，叶子结点上的样本数量，不纯度，最大叶子节点数等)进行剪枝，可以分为预剪枝和后剪枝，我们一般采用后剪枝的方法；
(2) 利用K -folds交叉验证，将训练集分为K份，然后进行K次交叉验证，每次使用K − 1 份作为训练样本数据集，另外一份作为测试集；
(3) 减少特征数量，剔除低相关特征，通过L1正则化进行特征选择。
(4) 剔除异常值。

1.3.15 决策树需要进行归一化处理吗？

决策树是一种简单高效并且具有强解释性的**概率模型**，是给定特征条件下所属类别的条件概率分布的一种退化表示(虽然它并不求解概率密度函数，但依据训练样本可以用类别频率来代替概率)。

概率模型是不需要归一化的，概率模型不关心变量的值，而是关心变量的分布和变量之间的条件概率。数值缩放不会影响决策树的分裂点位置。

1.3.16 常用的决策树一定是二叉树吗？二叉决策树与多分支决策树相比各有什么特点？

不一定，C4.5和ID3都是多叉树。

二叉决策树不像多叉树那样会形成过多的**数据碎片**，而二叉决策树可能会得到**更深**的最终决策树。

1.3.17 在一棵决策树构建过程中较为耗时的步骤是什么？

确定最佳分割点，在该步骤中需要对特征值进行排序，计算信息增益等，非常耗时。

1.3.18 举例说明什么时候其他模型比决策树会有更好的表现

决策树算法主要应用于非线性模型，若已知数据集是满足线性假设的，那么我们可以直接用一个线性模型取得比较好的预测结果。

1.3.19 决策树在选择特征进行分类时一个特征被选择过后，之后还会选择到这个特征吗？

ID3和C4.5的特征在层级之间不会重复，但CART是会复用的。

1.3.20 决策树模型的优缺点

优点：
① 易于理解，决策树易于理解和实现。人们在通过解释后都有能力去理解决策树所表达的意义。
② 数据处理简单，决策树能够直接处理自变量的缺失值。其他的算法模型往往要求先把数据一

般化，比如去掉多余的或者空白的属性。
③ 能够同时处理数据型变量和类别变量。其他的技术往往要求特征是单一类型的。
④ 是一个白盒模型，如果给定一个观察的模型，那么根据所产生的决策树很容易推出相应的逻辑表达式。
⑤ 易于通过静态测试来对模型进行评测。表示有可能测量该模型的可信度。
⑥ 在相对短的时间内能够对大型数据源做出可行且效果良好的结果。
⑦ 如果有不相关的 feature，没什么干扰，如果数据中有不相关的 feature，这个 feature 可能不出现在树的节点里。逻辑回归和 SVM 没有这样的天然特性(但是有相应的解决方案，比如逻辑回归里的 L1 正则化)。
⑧ 决策树可以用作变量选择的工具。
⑨ 决策树只使用了定序或连续自变量取值的大小顺序，它对自变量的测量误差或异常值是稳健的。

缺点：
① 很容易在训练数据中生成复杂的树结构，造成过拟合。剪枝可以缓解过拟合。
② 不适合处理高维数据，当属性数量过大的时候，部分决策树就不太适用了。
③ 每个非叶节点的划分都只考虑单个变量，因此很难发现基于多个变量的组合的规则。
④ 为每个非叶节点选择最优划分时，都仅考虑对当前节点划分的结果，这样只能够达到局部最优，而无法达到全局最优。
⑤ 由于决策树的贪心特性，树的结构不稳定，往往一个样本的输入会导致树结构很大的变动。
⑥ 无法解决异或问题，而神经网络可以解决该问题。
⑦ 如果某些特征的样本比例过大，生成的决策树容易偏向于这些特征。可以通过调节样本权重来改善。

1.3.21 树模型的应用场景

① 如果不强调模型的解释度，尽量避免单棵决策树，用集成树模型。
② 在集成树模型中，优先推荐使用XGBoost。
③ 在中小数据集上，优先选择集成树模型。大数据集上推荐神经网络。
④ 在需要模型解释度的项目上，优先使用树模型。
⑤ 在项目时间较短的项目上，如果数据质量低(大量缺失值、噪音等)，优先使用集成树模型。
⑥ 在硬件条件有限及机器学习知识有限的前提下，优先选择树模型。

1.3.22 决策树与逻辑回归的区别

① 决策树可以处理有**缺失值**的数据，而逻辑回归需要预先对缺失数据进行处理；

② 逻辑回归对数据**整体结构**的分析优于决策树，而决策树对**局部结构**的分析优于逻辑回归；(决策树由于采用分割的方法，所以能够深入数据内部，但同时失去了对全局的把握。一个分层一旦形成，它和别的层面或节点的关系就被切断了，以后的挖掘只能在局部中进行。同时由于切分，样本数量不断萎缩，所以无法支持对多变量的同时检验。而逻辑回归，始终着眼整个数据的拟合，所以对全局把握较好。但无法兼顾局部数据，或者说缺乏探查局部结构的内在机制)
③ 逻辑回归擅长分析**线性关系**，而决策树对线性关系的把握较差但对非线性数据拟合效果较好。线性关系在实践中有很多优点：简洁，易理解，可以在一定程度上防止对数据的过度拟合。
④ 逻辑回归对**异常值**比较敏感，容易受极端值的影响，而决策树对异常值具有稳健性。
⑤ 应用上的区别：**决策树的结果相比于逻辑回归略显粗糙**。逻辑回归原则上可以提供数据中每个样本点的概率，而决策树只能把挖掘对象分为有限的概率组群。比如决策树确定17个节点，全部数据就只能有17个概率，在应用上受到一定限制。就操作来说，决策树比较容易上手，需要的数据预处理较少，而逻辑回归则要去一定的训练和技巧。
⑥ **执行速度**上：当数据量很大的时候，逻辑回归的执行速度非常慢，而决策树的运行速度明显快于逻辑回归。

第二章 采样与集成

2.1 Bootstrapping

Bootstrapping 名字来自成语“pull up by your own bootstraps”，意思是依靠你自己的资源，称为自助法，它是一种有放回的抽样方法，是非参数统计中一种重要的估计统计量方差进而进行区间估计的统计方法。其核心思想和基本步骤如下：
(1) 采用重抽样技术从原始样本中抽取一定数量的样本，此过程允许重复抽样。
(2) 根据抽出的样本计算给定的统计量 T。
(3) 重复上述 N 次（一般大于1000），得到 N 个统计量 T。
(4) 计算上述 N 个统计量 T 的样本方差，得到统计量的方差。

应该说Bootstrap是现代统计学较为流行的一种统计方法，在小样本时效果很好。通过方差的估计可以构造置信区间等，其运用范围得到进一步延伸。

2.2 Bagging

Bootstrap aggregating的缩写。让该学习算法训练多轮，每轮的训练集由从初始的训练集中随机

取出的n个训练样本组成，某个初始训练样本在某轮训练集中可以出现多次或根本不出现，训练之后可得到一个预测函数序列 $h_1, \cdots, h_n$ ，最终的预测函数H对分类问题采用投票方式，对回归问题采用简单平均方法对新示例进行判别。(训练R个分类器 f_i，分类器之间只有参数不同。其中 f_i 是通过从N个训练样本中有放回地随机取N个样本构成的训练集上训练得到的。对于样本d，用这R个分类器去预测，得到最多的那个结果类别作为d的最终类别)

2.3 Boosting

其中主要的是AdaBoost (Adaptive Boosting)。初始化时对每一个训练样例赋相等的权重 $\frac{1}{n}$，然后用该学习算法对训练集训练t轮，每次训练后，对训练失败的训练例赋以较大的权重，也就是让学习算法在后续的学习中集中对比较难的训练例进行学习，从而得到一个预测函数序列 $h_1, \cdots, h_m$，其中 h_i 也有一定的权重，预测效果好的预测函数权重较大，反之权重较小。最终的预测函数H对分类问题采用有权重的投票方式，对回归问题采用加权平均的方法对新样例进行判别。(类似Bagging方法，但是训练是串行进行的(Bagging的训练是并行的)，第k个分类器训练时关注前k-1个分类器中错分的样例，即不是随机取，而是加大取这些样例的概率)

2.4 Bagging与Boosting的区别

二者的主要区别是取样方式不同。
(1) Bagging采用均匀取样，而Boosting根据错误率来取样，因此Boosting的分类精度要优于Bagging。
(2) Bagging的训练集的选择是随机的，各轮训练集之间相互独立，而Boosting的各轮训练集的选择与前面各轮的学习结果有关。
(3) Bagging的各个预测函数没有权重，而Boosting是有权重的。
(4) Bagging的各个预测函数可以并行生成，而Boosting的各个预测函数只能顺序生成。对于像神经网络这样极为耗时的学习方法，Bagging可通过并行训练节省大量时间开销。
(5) Bagging和Boosting都可以有效地提高分类的准确性。在大多数数据集中，Boosting的准确性比Bagging高。但在有些数据集中，Boosting会引起退化导致过拟合。
(6) Boosting思想的一种改进型**AdaBoost**方法在邮件过滤、文本分类方面都有很好的性能。
(7) Bagging方法使得方差减小，Boosting方法使得偏差减小。

2.5 Random Forest(RF)

随机森林是用随机的方式建立一个森林，森林由很多的决策树组成，随机森林的每一棵决策树之

间没有关联。在得到森林之后，当有一个新的输入样本进入时，森林中的每一棵决策树都会进行判断，预测这个样本所属的类别(对于分类算法)，将样本归为预测最多的类别。

2.5.1 Random Forest 生成步骤

step1从始训练数据集中，应用**bootstrap**方法有放回地随机抽取k个新的自助样本集，并由此构建k棵分类回归树，每次未被抽到的样本组成了K个袋外数据(out-of-bag)。
step2：设有n 个特征，则在每一棵树的每个节点处随机抽取m个特征，通过计算每个特征蕴含的信息量，从m个特征中选择一个最具有分类能力的特征进行节点分裂。
step3：每棵树最大限度地生长， 不做任何剪裁。
step4：将生成的多棵树组成随机森林， 用随机森林对新的数据进行分类， 分类结果按树分类器投票多少而定。

2.5.2 Random Forest 的优点

(1) 不必担心过度拟合；
(2) 适用于数据集中存在大量未知特征；
(3) 能够估计哪个特征在分类中更重要，可以将特征按重要性进行排序；
(4) 具有很好的抗噪声能力；
(5) 算法容易理解；
(6) 可以并行处理。

2.5.3 Random Forest 的缺点

(1) 对**小量数据集**和**低维数据集**的分类不一定可以得到很好的效果。
(2) 执行速度虽然比Boosting等快，但是比单个的决策树慢很多。
(3) 可能会出现一些差异度非常小的树，淹没了一些正确的决策。
(4) 由于树是随机生成的，**结果不稳定**(kpi值比较大)。

2.5.4 为什么 Random Forest 不会发生过拟合？

在建立每一棵决策树的过程中，有两点需要注意：**采样**与**完全分裂**。

首先是两个随机采样的过程，Random Forest对输入的数据要实施行、列的采样。对于行采样，采用有放回的方式，也就是在采样得到的样本集合中，可能有重复的样本。假设输入样本为N个，那么采样的样本也为N个。这样使得在训练的时候，每一棵树的输入样本都不是全部的样本，使得相对不容易出现over-fitting。然后进行列采样，从M个feature中，选择m个(m << M)。

之后对采样之后的数据使用完全分裂的方式建立出决策树，这样决策树的某一个叶子节点要么是无法继续分裂的，要么里面的所有样本的都是指向的同一个分类。一般的决策树算法都一个重要的步骤 -- 剪枝，但是Random Forest不同，因为之前的两个随机采样的过程保证了随机性，所以就算不剪枝，也不会出现over-fitting。

2.5.5 Random Forest 与SVM 比较

(1) 不需要调节过多的参数，因为随机森林只需要调节树的数量，而且树的数量一般是越多越好，而其他机器学习算法，比如SVM，有非常多超参数需要调整，如选择最合适的核函数，正则惩罚等。
(2)分类较为简单、直接。随机森林和支持向量机都是非参数模型（复杂度随着训练模型样本的增加而增大）。相较于一般线性模型，就计算消耗来看，训练非参数模型更为耗时耗力。分类树越多，需要消耗更多时间来构建随机森林模型。同样，我们训练出来的支持向量机有很多支持向量，最坏情况为，我们训练集有多少实例，就有多少支持向量。虽然，我们可以使用多类支持向量机，但传统多类分类问题的执行一般是one-vs-all（所谓one-vs-all 就是将binary分类的方法应用到多类分类中。比如我想分成K类，那么就将其中一类作为positive），因此我们还是需要为每个类训练一个支持向量机。相反，决策树与随机森林则可以毫无压力解决多类问题。
(3) 随机森林比较容易入手实践。随机森林在训练模型上要更为简单。很容易可以得到一个又好且具鲁棒性的模型。随机森林模型的复杂度与训练样本和树成正比。支持向量机则需要我们在调参方面做些工作，除此之外，计算成本会随着类增加呈线性增长。
(4) 小数据上，SVM优异，而随机森林对数据需求较大。就经验来说，我们更愿意认为支持向量机在存在较少极值的小数据集上具有优势。随机森林则需要更多数据但一般可以得到非常好的且具有鲁棒性的模型。

2.5.6 Random Forest与Bagging的区别

(1) Random Forest是选与输入样本的数目相同多(bootstraps)的次数(可能一个样本会被选取多次，同时也会造成一些样本不会被选取到)，而Bagging一般选取比输入样本的数目少的样本；
(2) Bagging是用全部特征来得到分类器，而Random forest是需要从全部特征中选取其中的一部分来训练得到分类器；
(3) 一般Random Forest的效果比Bagging效果好！

2.5.8 Random Forest与Gradient Boosting Decision Tree(GBDT)区别

决策树+bagging=随机森林;

决策树+Boosting=GBDT。

2.6 GBDT 及其改进

2.6.1 Gradient Boosting Decision Tree (GBDT)

Boosting是一种思想，Gradient Boosting是一种实现Boosting的方法，它主要的思想是，每一次建立模型是在之前建立模型**损失函数的梯度下降方向**。损失函数(loss function)描述的是模型的不靠谱程度，损失函数越大，则说明模型越容易出错。如果我们的模型能够让损失函数持续的下降，则说明我们的模型在不停的改进，而最好的方式就是让损失函数在其梯度（Gradient)的方向上下降。

2.6.2 GBDT 为什么使用负梯度来代替残差

(1) 残差是使用平方损失函数时的特殊情况，倘若使用其他损失函数，那么再沿着残差的方向进行训练，模型的效果就不会再逐次提升了，而负梯度可以很好的拓展到任何一阶可导的损失函数中。
(2) 负梯度是损失函数下降最快的方向。
(3) 一般来说，损失函数中不光有残差这类基本损失项(例如残差平方和)，还包含正则化项，有时候虽然基本损失项在下降，而正则化项却在上升，从而导致整体损失在上升。负梯度是整体损失的共同导数，它可以保证整体损失保持下降。

2.6.3 XGBoost 为什么使用二阶梯度展开

(1) 二阶梯度能够使得梯度收敛的更快更准确；
(2) 可以支持自定义损失函数

2.6.4 LightGBM相较于XGBoost做了哪些改进？

(1) 为了使得XGB能够支持并行运算，XGB对数据进行了预排序，而这个过程是极其耗时的，LGB通过使用直方图算法加速了这一处理过程。
(2) 通过二阶梯度进行单边梯度采样。
(3) 互斥特征合并。
(4) XGB不支持处理分类特征，LGB支持处理分类特征。

2.6.5 LightGBM 是怎么输出概率值的？

(1)对于二分类问题：假设LightGBM模型公训练K轮，每一轮的输出结果都是一个权值W_k(实数域取值)，记$W = \sum_{k=1}^{K} W_k$，则通过对W进行sigmoid变换即可得到预测概率值$P\{Y = 1\} = \frac{1}{1+e^W}$。

(2)对于多分类问题：LightGBM会将其拆分为多个二分类问题，然后通过(1)中的方式计算实例属于每个类别的概率值，然后再通过softmax输出实例所属类别的概率值。

2.6.6 CatBoost 做了哪些工作？

参考：https://zhuanlan.zhihu.com/p/102540344

2.6.7 Boosting树模型中有哪些可调节的参数？

树的深度depth，叶子节点数num_leaf，列抽样比例，L1正则化系数，L2正则化系数，训练轮次。

2.6.8 XGBoost，Lightgbm，CATBoost的生长策略

XGB：level-wise;

LGB：leaf-wise;

CAT：SymmetricTree。

第三章 Naive Bayesian (NB)

3.1 基本知识

3.1.1 先验概率

3.1.2 条件概率

3.1.3 后验概率

3.1.4 误判损失

3.1.5 条件风险

3.2 面试题

3.2.1什么是朴素贝叶斯

朴素贝叶斯算法（Naive Bayes, NB) 是应用最为广泛的分类算法之一。它是基于贝叶斯定理和特征条件独立假设的分类器方法。由于朴素贝叶斯法基于贝叶斯公式计算得到，有着坚实的数学基础，以及稳定的分类效率。NB模型所需估计的参数很少，对缺失数据不太敏感，算法也比较简单。以前的垃圾邮件分类都是基于朴素贝叶斯分类器识别的。

原理：朴素贝叶斯是基于贝叶斯定理与特征条件独立假设的分类方法。对于给定的待分类项x，通过学习到的模型计算后验概率分布，即：在此项出现的条件下各个目标类别出现的概率，将后验概率最大的类作为x所属的类别

3.2.2 怎么理解朴素贝叶斯中的“朴素”？

朴素贝叶斯有一个很强的假设，即所有特征之间都是相互独立的，他们与目标变量的预测作用是相同的。而在现实中，特征之间往往都是存在相关性的，因而这个假设很天真，很朴素。

3.2.3 朴素贝叶斯的工作流程

朴素贝叶斯的工作流程可以分为三个阶段：准备阶段、分类器训练阶段和应用阶段。

准备阶段：这个阶段的任务是为朴素贝叶斯分类做必要的准备，主要工作是根据具体情况确定特征属性，并对每个特征属性进行适当划分，去除高度相关性的属性(如果两个属性具有高度相关性的话，相当于该属性将会在模型中发挥了2次作用，会使得朴素贝叶斯所预测的结果向该属性所希望的方向偏离，导致分类出现偏差)，然后由人工对一部分待分类项进行分类，形成训练样本集合。这一阶段的输入是所有待分类数据，输出是特征属性和训练样本。(这一阶段是整个朴素贝叶斯分类中唯一需要人工完成的阶段，其质量对整个过程将有重要影响)

分类器训练阶段：这个阶段的任务就是生成分类器，主要工作是计算每个类别在训练样本中的出现频率及每个特征属性划分对每个类别的条件概率估计，并将结果记录。其输入是特征属性和训练样本，输出是分类器。这一阶段是机械性阶段，根据公式可以由程序自动计算完成。

应用阶段：这个阶段的任务是使用分类器对待分类项进行分类，其输入是分类器和待分类项，输出是待分类项与类别的映射关系。这一阶段也是机械性阶段，由程序完成。

3.2.4. 朴素贝叶斯的优缺点

优点：
(1) 朴素贝叶斯模型发源于古典数学理论，有着坚实的数学基础，以及稳定的分类效率。
(2) 对大数量训练和查询时具有较高的速度。即使使用超大规模的训练集，针对每个项目通常也只会有相对较少的特征数，并且对项目的训练和分类也仅仅是特征概率的数学运算而已；
(3) 对小规模的数据表现很好，能个处理多分类任务，适合增量式训练（即可以实时的对新增的样本进行训练）；
(4) 对缺失数据不太敏感，算法也比较简单，常用于文本分类；
(5) 朴素贝叶斯对结果解释容易理解。

缺点：
(1) 需要计算先验概率。
(2) 理论上，朴素贝叶斯模型与其他分类方法相比具有最小的误差率。但是在实际中，因为朴素贝叶斯“朴素，”的特点，导致在属性个数比较多或者属性之间相关性较大时，分类效果不好。而在属性相关性较小时，朴素贝叶斯性能最为良好。
(4) 对训练数据的依赖性很强，如果训练数据误差较大，那么预测出来的效果就会不佳。
(4) 由于**使用了样本属性独立性的假设，因此，当样本属性有关联时其效果不好**。可以使用数据降维(PCA等)的方法，去除特征相关性，再进行朴素贝叶斯计算，但可解释性会变差。

3.2.5 为什么朴素贝叶斯存在“朴素”的缺点假设，仍然可以取得较好的预测效果？

对于分类任务来说，只要各个条件概率之间的排序正确，那么就可以通过比较概率大小来进行分类，不需要知道精确的概率值(朴素贝叶斯分类的核心思想是找出后验概率最大的那个类，而不是求出其精确的概率)。

如果属性之间的相互依赖对所有类别的影响相同，或者相互依赖关系可以互相抵消，那么属性条件独立性的假设在降低计算开销的同时不会对分类结果产生不良影响。

3.2.6 朴素贝叶斯为什么做独立性假设？

3.2.7 什么是拉普拉斯平滑法?

拉普拉斯平滑法是朴素贝叶斯中处理零概率问题的一种修正方式。在进行分类的时候，可能会出现某个属性值在训练集中没有与某个类同时出现过的情况，如果直接基于朴素贝叶斯分类器的表达式进行计算就会出现零概率。为了避免其他属性所携带的信息被训练集中未出现过的属性值“抹去”，所以才使用拉普拉斯估计器进行修正。

具体的方做法是：先在分子上加1；对于先验概率，在分母上加上训练集中可能的类别数；对于条件概率，则在分母上加上第i个属性可能的取值数。

3.2.8 朴素贝叶斯中有没有超参数可以调？

朴素贝叶斯是没有超参数可以调的，所以它不需要调参，朴素贝叶斯是根据训练集进行分类，分类出来的结果基本上就是确定了的，拉普拉斯估计器不是朴素贝叶斯中的参数，不能通过拉普拉斯估计器来对朴素贝叶斯调参。

3.2.9 朴素贝叶斯中有多少种模型？

朴素贝叶斯含有3种模型：(1) 高斯模型，对连续型数据进行处理；(2) 多项式模型，对离散型数据进行处理，计算数据的条件概率(使用拉普拉斯估计器进行平滑的一个模型)；(3) 伯努利模型，伯努利模型的取值特征是布尔型，即出现为ture,不出现为false，在进行文档分类时，就是一个单词有没有在一个文档中出现过。

3.2.10 朴素贝叶斯有哪些应用？

朴素贝叶斯的应用最广的应该就是在文档分类、垃圾文本过滤(如垃圾邮件、垃圾信息等)、情感分析(微博、论坛上的积极、消极等情绪判别)这些方面，除此之外还有多分类实时预测、推荐系统(贝叶斯与协同过滤组合使用)、拼写矫正(当你输入一个错误单词时，可以通过文档库中出现的概率对你的输入进行矫正)等。

3.2.11 朴素贝叶斯对异常值是否敏感？

朴素贝叶斯对异常值不敏感。所以在进行数据处理时，我们可以不去除异常值，因为保留异常值可以保持朴素贝叶斯算法的整体精度，而去除异常值则可能在进行预测的过程中由于失去部分异常值导致模型的泛化能力下降。

3.2.12 朴素贝叶斯是高方差还是低方差模型？

朴素贝叶斯是低方差模型。(误差 = 偏差 + 方差)对于复杂模型来说，由于复杂模型充分拟合了部分数据，使得它们的偏差变小，但由于对部分数据过分拟合，这就导致预测的方差会变大。因为朴素贝叶斯假设了各个属性之间是相互的，算是一个简单的模型。对于简单的模型来说，则恰恰相反，简单模型的偏差会更大，相对的，方差就会较小。(偏差是模型输出值与真实值的误差，也就是模型的精准度，方差是预测值与模型输出期望的的误差，即模型的稳定性，也就是数据的集中性的一个指标)

3.2.13 素贝叶斯与LR的区别？

(1) 朴素贝叶斯是生成模型，根据已有样本进行贝叶斯估计学习出先验概率P(Y)和条件概率P(X|Y)，进而求出联合分布概率P(X,Y)，最后利用贝叶斯定理求解P(Y|X)， 而LR是判别模型，根据极大化对数似然函数直接求出条件概率P(Y|X)；
(2) 朴素贝叶斯是基于很强的条件独立假设（在已知分类Y的条件下，各个特征变量取值是相互独立的），而LR则对此没有要求；
(3) 朴素贝叶斯适用于数据集少的情景，而LR适用于大规模数据集。

前者是生成式模型，后者是判别式模型，二者的区别就是生成式模型与判别式模型的区别。

(1) 首先，Navie Bayes通过已知样本求得先验概率P(Y)，及条件概率P(X|Y)，对于给定的实例，计算联合概率，进而求出后验概率。也就是说，它尝试去找到底这个数据是怎么生成的（产生的），然后再进行分类。哪个类别最有可能产生这个信号，就属于那个类别。

优点：
① 样本容量增加时，收敛更快；
② 隐变量存在时也可适用。

缺点：
① 时间长；
② 需要样本多；
③ 浪费计算资源。

(2) 相比之下，Logistic回归不关心样本中类别的比例及类别下出现特征的概率，它直接给出预测模型的式子。设每个特征都有一个权重，训练样本数据更新权重w，得出最终表达式。

优点：
① 直接预测往往准确率更高；
② 简化问题；
③ 可以反应数据的分布情况，类别的差异特征；
④ 适用于较多类别的识别。

缺点：
① 收敛慢；
② 不适用于有隐变量的情况。

第四章 Logistic Regression (LR)

注：参考 https://zhuanlan.zhihu.com/p/74874291

4.1 Logistic 分布

Logistic 分布是一种连续型的概率分布，其分布函数和密度函数分别为：

$$F(x) = P(X \leq x) = \frac{1}{1+e^{-(x-\mu)/\gamma}}$$

$$f(x) = F'(x) = \frac{e^{-(x-\mu)/\gamma}}{\gamma(1+e^{-(x-\mu)})^2}$$

其中μ表示位置参数，$\gamma > 0$为形状参数；
Logistic的分布域正太分布相似，但尾部更长，波峰更高；

sigmoid函数是Logistic分布函数在$\mu = 0, \gamma = 1$时的特殊情形。

4.2 LR与其他模型对比

4.2.1 LR与线性回归

联系：
① 逻辑回归是在线性回归的基础上加了一个 Sigmoid 函数（非线形）映射，使得逻辑回归称为了一个优秀的分类算法；
② 本质上来说，两者都属于广义线性模型，但他们两个要解决的问题不一样，逻辑回归解决的是分类问题，输出的是离散值，线性回归解决的是回归问题，输出的连续值

区别：
① 线性回归是在实数域范围内进行预测，而分类范围则需要在 [0,1]，逻辑回归减少了预测范围；
② 线性回归在实数域上敏感度一致，而逻辑回归在 0 附近敏感，在远离 0 点位置不敏感，这个的好处就是模型更加关注分类边界，可以增加模型的鲁棒性。

4.2.2 LR与SVM

相同点：
① 都是分类算法，本质上都是在找最佳分类超平面；
② 都是监督学习算法；
③ 都是判别式模型，判别模型不关心数据是怎么生成的，它只关心数据之间的差别，然后用差别来简单对给定的一个数据进行分类；
④ 都可以增加不同的正则项。

不同点：
① LR 是一个统计的方法，SVM 是一个几何的方法；
② SVM 的处理方法是只考虑 Support Vectors，也就是和分类最相关的少数点去学习分类器。而逻辑回归通过非线性映射减小了离分类平面较远的点的权重，相对提升了与分类最相关的数据点的权重；
③ 损失函数不同：LR 的损失函数是交叉熵，SVM 的损失函数是 HingeLoss，这两个损失函数的目的都是增加对分类影响较大的数据点的权重，减少与分类关系较小的数据点的权重。对 HingeLoss 来说，其零区域对应的正是非支持向量的普通样本，从而所有的普通样本都不参与最终超平面的决定，这是支持向量机最大的优势所在，对训练样本数目的依赖大减少，而且提高

了训练效率；
④ LR 是参数模型，SVM 是非参数模型，参数模型的前提是假设数据服从某一分布，该分布由一些参数确定（比如正太分布由均值和方差确定），在此基础上构建的模型称为参数模型；非参数模型对于总体的分布不做任何假设，只是知道总体是一个随机变量，其分布是存在的（分布中也可能存在参数），但是无法知道其分布的形式，更不知道分布的相关参数，只有在给定一些样本的条件下，能够依据非参数统计的方法进行推断。所以 LR 受数据分布影响，尤其是样本不均衡时影响很大，需要先做平衡，而 SVM 不直接依赖于分布；
⑤ LR 不依赖样本之间的距离，SVM 是基于距离的；
⑥ LR 相对来说模型更简单好理解，特别是大规模线性分类时并行计算比较方便。而 SVM 的理解和优化相对来说复杂一些，SVM 转化为对偶问题后，分类只需要计算与少数几个支持向量的距离，这个在进行复杂核函数计算时优势很明显，能够大大简化模型和计算。

4.2.3 LR与NB

相同点：
朴素贝叶斯和逻辑回归都属于分类模型，当朴素贝叶斯的条件概率 [公式] 服从高斯分布时，它计算出来的 P(Y=1|X) 形式跟逻辑回归是一样的。

不同点：
① 逻辑回归是判别式模型 p(y|x)，朴素贝叶斯是生成式模型 p(x,y)：判别式模型估计的是条件概率分布，给定观测变量 x 和目标变量 y 的条件模型，由数据直接学习决策函数 y=f(x) 或者条件概率分布 P(y|x) 作为预测的模型。判别方法关心的是对于给定的输入 x，应该预测什么样的输出 y；而生成式模型估计的是联合概率分布，基本思想是首先建立样本的联合概率概率密度模型 P(x,y)，然后再得到后验概率 P(y|x)，再利用它进行分类，生成式更关心的是对于给定输入 x 和输出 y 的生成关系；
② 朴素贝叶斯的前提是条件独立，每个特征权重独立，所以如果数据不符合这个情况，朴素贝叶斯的分类表现就没逻辑回归好了。

4.3 模型细节

4.3.1 为什么适合离散特征

我们在使用逻辑回归的时候很少会把数据直接丢给 LR 来训练，我们一般会对特征进行离散化处理，这样做的优势大致有以下几点：

① 离散后稀疏向量内积乘法运算速度更快，计算结果也方便存储，容易扩展；

离散后的特征对异常值更具鲁棒性，如 age>30 为 1 否则为 0，对于年龄为 200 的也不会对模型造成很大的干扰；
② LR 属于广义线性模型，表达能力有限，经过离散化后，每个变量有单独的权重，这相当于引入了非线性，能够提升模型的表达能力，加大拟合；
③ 离散后特征可以进行特征交叉，提升表达能力，由 M+N 个变量编程 M*N 个变量，进一步引入非线形，提升了表达能力；
④ 特征离散后模型更稳定，如用户年龄区间，不会因为用户年龄长了一岁就变化；

总的来说，特征离散化以后起到了加快计算，简化模型和增加泛化能力的作用。

4.3.2 为什么不使用平方误差

假设目标函数为MSE，即：
$L = \frac{(y-\hat{y})^2}{2}$
$\frac{\partial L}{\partial w} = (\hat{y} - y)\sigma^{'}(w \cdot x)x$
Sigmoid的导数项为：
$\sigma^{'}(w \cdot x) = w \cdot x(1 - w \cdot x)$
根据 w 的初始化，导数值可能很小（想象一下 Sigmoid 函数在输入较大时的梯度）而导致收敛变慢，而训练途中也可能因为该值过小而提早终止训练（梯度消失）。
另一方面，交叉熵的梯度如下，当模型输出概率偏离于真实概率时，梯度较大，加快训练速度，当拟合值接近于真实概率时训练速度变缓慢，没有 MSE 的问题。
$g^{'} = \sum_{i=1}^{N} x_i(y_i - p(x_i))$

4.3.3 适用条件

二分类logistic回归需要满足以下6个条件：

条件1：因变量为二分类变量。
条件2：至少有1 个自变量，可以是分类变量，也可以是连续变量。
条件3：因变量的观察结果相互独立。
条件4：例数较少类的因变量例数为自变量个数的10~15 倍(EPV原则)，且经验上两组的人数最好>30例，自变量的参照水平组不应少于30或50例。
条件5：自变量之间无多重共线性。
条件6：自变量不存在明显的异常值。

第五章 Support Vector Machine (SVM)

注：参考 https://zhuanlan.zhihu.com/p/49331510

5.1 SVM分类

5.1.1 线性可分SVM（硬间隔SVM）

5.1.2 线性SVM（软间隔SVM）

5.1.1 非线性SVM（核方法）

5.2 SVM优缺点

优点：
① 由于SVM是一个凸优化问题，所以求得的解一定是全局最优而不是局部最优。
② 不仅适用于线性线性问题还适用于非线性问题(用核技巧)。
③ 拥有高维样本空间的数据也能用SVM，这是因为数据集的复杂度只取决于支持向量而不是数据集的维度，这在某种意义上避免了“维数灾难”。
④ 理论基础比较完善(例如神经网络就更像一个黑盒子)。

缺点：
① 二次规划问题求解将涉及m阶矩阵的计算(m为样本的个数), 因此SVM不适用于超大数据集。(SMO算法可以缓解这个问题)
② 只适用于二分类问题。(SVM的推广SVR也适用于回归问题；可以通过多个SVM的组合来解决多分类问题)

第六章 Factorization Machine (FM)

6.1 FM的优势

① 适用于极端稀疏的输入；

② 线性的计算复杂度支持海量数据训练；
③ 适用范围广。

6.2 MF（Matrix Factorization）与FM的关系

本质上，MF模型是FM模型的特例，MF可以被认为是只有User ID 和Item ID这两个特征Fields的FM模型，MF将这两类特征通过矩阵分解，来达到将这两类特征embedding化表达的目的。而FM则可以看作是MF模型的进一步拓展，除了User ID和Item ID这两类特征外，很多其它类型的特征，都可以进一步融入FM模型里，它将所有这些特征转化为embedding低维向量表达，并计算任意两个特征embedding的内积，就是特征组合的权重，如果FM只使用User ID 和Item ID，它的预测过程和MF的预测过程是一样的

第七章 无监督学习方法

谱聚类，k-means，dbscan，mean-shift，gmm

第八章 因果推断

8.1实例

幸存者偏差，因普森悖论

8.2 因果推断分类

8.2.1 因果效应推断

1、二值因果效应推断；
2、多值因果效应推断；
3、连续值因果效应推断；
4、随时间变化的因果效应推断。

8.2.2 因果关系推断

8.3 框架

潜在因果框架（RCM）；
因果图框架（SCM）

8.4 因果推断的三个层级

1、关联问题：观测到A，B会怎样？
2、干预问题：改变A，B会怎样？
3、反事实推断：假如没有A，B会怎样？

8.5 因果推断的应用场景

8.5.1 A/B实验

1、产品A/B实验；
2、策略A/B实验；
3、模型A/B实验。

8.5.2 伪A/B实验

1、PSM；
2、IPW；
3、DID；

8.5.3 Uplift模型

1、红包；
2、广告。

8.5.4 因果关系

关系挖掘

8.6 相关算法

因果森林，正交随机森林

第3部分 面试现场

注：(1)为判断题目出现频率及题目难度，保留重复记录；
(2)题目无解析；
(3)部分题目记录来源公司、岗位；
(4)只记录算法相关问题；
(5)项目相关问题占比较大，但这类问题具有特殊性，只保留个别有关项目的典型问题；
(6)不区分一、二面；
(7)只记录技术面，不保留HR面；
(8)有些问题已经忘记了，只记录了一小部分，因此以下问题不代表全部。

快手（数据挖掘算法工程师--日常实习）

1、lgb相对于xgb做了哪些改进？
① 直方图加速；
② 单边梯度采样；
③ 互斥特征合并；
④ 并行计算；
⑤ 支持分类特征处理。

2、了解哪些分类算法？
perceptron, lr, knn, nb, svm, rf, adaboost, gbdt, xgb, lgb, catboost.

3、决策树的节点分裂准则是什么？存在什么问题？
信息增益，偏向选择取值较多的特征

4、L1正则化为什么能够使得系数向量(矩阵)稀疏？
参数的可行域与约束域交在坐标轴上

5、(编程)两个栈实现一个队列。

6、(编程)有序数组实现二分查找。

7、尝试过哪些特征交叉、特征组合方法。

8、特征工程的作用？为什么进行特征选择？
① 减少计算量
② 避免过拟合
③ 提升模型能力上限

9、(编程)查找数组中的主要元素(占比超过一半)。

10、O(1)空间复杂度优化9中的方法。

注：两轮技术面试，每次35分钟左右

字节（推荐算法实习生--日常实习）

1、相似性度量（如欧式距离、曼哈顿距离，余弦相似度）之间的区别？

2、（编程）求四次方根。

注：项目、比赛等问了15个问题左右，共计80分钟左右

滴滴（ETA算法与路况策略）

1、stacking模型融合方法。

2、lstm模型。

3、GBDT中，输入N*M的数据（N特征，M数据量），导数的维度是多少？
N*1

4、XGB、lightGBM分别作了哪些改进工作？
XGB：用进行了二阶梯度展开，提升了准确率和收敛速度；为进行并行训练加速，对数据进行预排序。
lgb见上面

5、树模型的Boosting方法中是怎么解决过拟合问题的？
① 限制树的深度
② 叶子节点数
③ 迭代轮次
④ 增加正则化

⑤ 设定节点分裂的最小阈值
⑥ 限制不纯度

6、深度学习中是怎么解决过拟合的。
① 选择合适的batch size
② 进行batch normal
③ 使用随机优化方法
④ 使用dropout策略
⑤ 施加动量momentum

7、XGB的基模型是什么？
cart回归树

8、缺失值填充方法？
① 0填充
② 均值填充
③ 众数填充
④ knn填充
⑤ 回归填充

9、Boosting和Bagging方法那个解决方差，那个解决偏差？
boosting-偏差，bagging-方差

10、（编程）最大子序和。

11、10中方法的时间复杂度。

注：约40分钟

斯伦贝谢（AI Intern）

1、什么是假设检验？

2、假设检验的流程。
① 选择显著性水平α
② 确定检验统计量
③ 确定样本量
④ 采样

⑤ 计算临界值
⑥ 和临界值做比较
⑦ 决策

3、什么是t检验？

4、假设检验的p值怎么确定？
常用值：0.1，0.05，0.01

5、如何估计某地区pizza店的数量？
回归问题

6、lightGBM是怎么输出概率值的？
将每轮预测得到的权值相加，然后通过sigmoid运算即得概率值

注：两个面试官同时面，需要英语自我介绍，总计30分钟左右

京东（广告）

1、lightGBM有哪些可以调整的参数？
① 最大深度
② 最大叶子节点数
③ 列采样比例
④ L1正则化系数
⑤ L2正则化系数
⑥ 迭代轮次

2、参数比较多时，你有什么好的调参策略，或则有什么标准可以参考？
固定其他参数，每次调整其中的两个

3、lightgbm中怎么避免过拟合？
① 限制深度
② 限制叶子节点数
③ 调整列抽样比例
④ 施加正则化
⑤ 减少迭代轮次

4、深度学习中怎么避免过拟合？（见前面）

5、CNN，RNN，Transformer。

6、（编程）二分查找。

7、（编程）排序算法（最好快排/堆排/归并）。

8、（编程）SQL开窗函数：一张表(table)中有字段"部门(department)"、"员工名(name)"、"员工薪资(salary)"，查找每个部门中薪资最高的员工的薪资、所在部门以及该员工的姓名。

9、（编程）最长回文序列。

注：两轮技术面，每次70分钟左右，编程一般两道middle难度或者一道easy+一道middle。

腾讯（pcg-腾讯新闻-Data Science-应用研究-暑期）

1、第一类错误和第二类错误。

2、p值的含义。
不拒绝原假设的概率。

3、增加样本量减少的是哪类错误的概率？
第二类错误。

4、在A/B test时，怎么处理的自然分布不一致的情况（比如一个自然周期为一周，性别分布，年龄分布，AB测试的周期、人群怎么定）？

5、怎么确定A/B test的用户量？

6、怎么解决用户冷启动问题（新用户没有历史数据特征）？

7、（编程）一枚质地均匀的硬币出现两次正面需要抛掷的次数的期望，编程怎么实现（蒙特卡洛模拟）？

8、手机广告定量投放给1000个用户，怎么找到这1000个用户，需要哪些特征？

9、基于以上1000个用户，又有50张高额优惠券，应该投放给哪些用户，从算法层面上怎么解释？

10、二分类问题中，为什么使用交叉熵损失而不是MSE损失？

11、（编程）对于任意二叉树，怎么计算它的高度？

12、Random Forest 和 Lightgbm有什么区别？

注：三轮技术面试+一轮HR面，每次限制三十分钟之内，问题多偏向统计理论，要求较高（hadoop，spark，hive，scala，git，前、后端知识等），因为时间问题，编程题只问了思路，没让实现，第三轮比较仓促就问了一个问题。

Aibee（机器学习）

1、处理类别不平衡问题的方法。
① under sampling
② over sampling
③ tomek link
④ smote
⑤ class weighting

2、类别加权方法中，怎么分配权值。
① $\frac{1}{N_class}$
② $\log(N_{class})$
③ r_{class}^{δ}

3、当维度m大于数据个数n时，怎么拟合数据的分布，因为参数个数一般也会大于数据点个数，从而导致没办法估计参数。
① 先降维再估计
② 数据过采样
③ smote方法合成新数据

4、对于一个很小的数据集（几十个数据点的样子），在统计检验时效果不太显著，用什么方法处理可以使得结果变得显著？
① 交叉验证
② bootstrap
③ jacknife
④ smote
⑤ 拟合分布再生成数据等

5、在拟合一个分布之后，怎么检验这个分布拟合的好不好？
① 构造拟和分布的$1-\alpha$置信区间，然后检验真实数据落在这个区间里的概率；
② 依真实数据构造一个验证区间，用拟合的分布生成新数据点，检验新数据点落在验证区间的

概率；
概率值越大，拟合的分布效果越好）

6、一枚硬币，正面朝上的概率是0.7，反面朝上的概率是0.3，连续抛掷1000次，求分布？
$X \sim B(1000, 0.7)$

7、基于6，记X为正面朝上的次数，求$P\{X > 700\}$。
0.5

8、怎么比较当前一个周期的收益和上一个周期的收益的差异。
拟合分布曲线，计算不同周期上的积分值进行比较

9、回归分析中，有些系数可能会非常小，那怎么判断这个系数实际上是不是应该为0？
① 对于一元回归分析，直接进行方程的显著性检验
② 对于多元回归方程来说，对系数进行相应的系数的显著性检验

10、回归分析的显著性检验中使用的统计量是哪个？
① 方程的显著性检验，使用F统计量
② 系数的显著性检验，使用t统计量

11、在推荐系统中，应该选取哪些数据作为基本特征，预测变量为该用户是否有购买该商品的意向？以某4s店为例。
人物画像特征：性别，年龄，画像人群，经济水平等级等
人物行为特征：到店次数，到店时长，是否有试用，试用时长，是否有业务交流，交流时长等
产品特征：产品类型，产品价格，产品热度，产品日期，产品历史评价（销量，好评，拉新等）

12、numpy的ndarray类型为什么比padas的dataframe类型查询速度更快？
numpy的ndarray是随机存储结构，在查询时可以按照索引以O(1)的时间复杂度获取，而pandas的dataframe类型在查询时，会按行进行逐列查询，最坏情况下的时间复杂度为O(m*n)

注：两轮技术面，每次30分钟左右，没有考察编程。

中电达通（算法实习生）

1、了解哪些机器学习算法，他们有什么特点？
perceptron, lr, knn, nb, svm, rf, adaboost, gbdt, xgb, lgb, catboost.

2、floyed算法和dijkstra算法。
计算最短路的算法

注：一轮技术面（约25分钟）+一轮笔试，偏运筹优化、专利申请。

金科览智（NLP）

1、贝叶斯估计。

2、什么是共轭分布？

3、（编程）排序问题（topk）。

注：一轮技术面，约20分钟。

格灵深瞳（CV）

1、numpy的ndarray类型为什么比padas的dataframe类型查询速度更快？
见前面

2、常用的聚类方法，以及他们的适用场景？
① k-means
② dbscan
③ mean-shift
④ canopy
⑤ gauss mix model

3、（编程）并查集。

4、（编程）查找最佳切分点，使得MSE最小。

中电金信（NLP）

1、词嵌入方法。
① one-hot
② tf-idf
③ word2vec
· · ·

2、lightgbm是怎么输出概率值的？
见前面

3、（编程）链表是否有环

龙湖集团（机器学习-校招）

1、ROC-AUC计算原理
ROC曲线：横轴为FPR（假阳率），纵轴为TPR（真阳率）。
AUC为ROC曲线下方围成的面积

记负样本量为N，正样本数量为P，则
$FPR = FP/N$
$TPR = TP/P$
通过调整判为正例的阈值（例如取每个样本被预测为正例的概率值，从小到大排列后依次作为判别阈值），来获得一系列的（FPR，TPR）的数值对，然后将其绘制在图上，将各点连接，这个图就是ROC曲线图。

关于ROC-AUC的一些补充：
① ROC曲线中的4个点和1条线：
第一个点(0,1)，即FPR=0, TPR=1，这意味着FN=0，并且FP=0。这是一个完美的分类器，它将所有的样本都正确分类。
第二个点，(1,0)，即FPR=1，TPR=0，所有正例都被预测为负例，即所有正例都没被预测出来，这是一个最糟糕的分类器，因为它成功避开了所有的正确答案。
第三个点，(0,0)，即FPR=TPR=0，即FP =TP =0，可以发现该分类器预测所有的样本都为负样本。
第四个点（1,1），分类器实际上预测所有的样本都为正样本。
虚线y=x：这条对角线上的点其实表示的是一个采用随机猜测策略的分类器的结果。例如(0.5,0.5)，表示该分类器随机对于一半的样本猜测其为正样本，另外一半的样本为负样本。出现在右下角三角形中的任何分类器都比随机猜测更糟糕。

② 为什么使用ROC曲线，而不是其他什么指标？
ROC曲线有个很好的特性：当测试集中的正负样本的分布变化的时候，ROC曲线能够保持不变。在实际的数据集中经常会出现类不平衡现象，即负样本比正样本多很多（或者相反），而且测试数据中的正负样本的分布也可能随着时间变化。而在这种情况下，ROC曲线能够保持不变

2、Macro-F1，Micro-F1的差别
混淆矩阵

真实情况	预测结果

	正例	反例
正例	TP（真正例）	FN（假反例）
反例	FP（假正例）	TN（真反例）

F1-Score的计算公式：$F1 = \frac{2PR}{P+R}$
其中，P是查准率（precision），R是查全率（Recall），计算公式为：
$P = \frac{TP}{TP+FP}, R = \frac{TP}{TP+FN}$

F1-Score、Macro-F1、Micro-F1的区别：
使用场景：
① 对于多分类任务，我们需要对每个类别分别计算一个混淆矩阵；由于F1只针对二分类的任务，要考虑多分类问题的F1-Score，就要用到macro-F1与micro-F1。
② 当数据分布不平衡时（类别不平衡），适合使用micro-F1，它考虑每个类别的样例数量；同时，因为它考虑了每个类别的样例数量，样例数量较多的类多较大的影响micro-F1的值；
③ macro-F1不考虑每个类别中样例的数量，将每个类别同等看待，如果一个类别的precision和recall较高，那么这个类将会对macro-F1的值产生较大影响。

3、fpr计算原理

4、常用的分类评价指标
ROC-AUC，Cross-Entropy，Recall，Precision，F1-Score，Macro-F1，Micro-F1

5、XGB怎么处理多分类问题
将多分类问题分成多个二分类问题，对每个二分类问题进行预测，然后对每个类别的预测结果进行softmax，取概率最大的类别作为最终预测结果

6、随机森林和XGB的区别
① 随机森林是bagging集成方法（并行运算），XGB是boosting集成方法（串行运算）；
② 随机森林不会过拟合，因为同时采用了行采样、列采样和投票决策，XGB相对而言容易过拟合，需要调整训练的轮次、树的深度、叶子节点数、L1/L2正则化系数；
③ 随机森林拟合方差，XGB拟合偏差；
④ 随机森林鲁棒性更强，XGB鲁棒性相较而言较弱

7、bagging是有放回还是无放回？为什么？
有放回。
首先我们假设无放回，看看会发生什么：

① 候选样本逐渐减少，基分类器数量受限，因为样本总量 = 每个基分类器的训练样本数 * 基分类器数量；即若要增加基分类器数量，那么每个基分类器的训练样本就会减少，导致分类器欠拟合；而如果要增加每个基分类器的训练样本数，那么基分类器数量就会减少，投票结果的可信度就会降低；
② 训练的基分类器可能并不能正常地进行预测，例如，前几个基分类器已经将具有代表性的样本全部选取，那么剩余的基分类器所使用的样本代表性较低；从而导致在投票时，结果偏差较大，集成的分类器效果较差；

8、lightgbm怎么处理类别不平衡
正样本权重=负样本数量/正样本数量
负样本权重=正样本数量/负样本数量

9、常用的类别不平衡处理方法

10、怎么防止XGB过拟合

11、XGB叶子节点数和树深度的关系，依据是什么？
二叉树

海尔（数据挖掘分析师-校招）

1、ROC-AUC的纵、横轴分别是什么
FPR，TPR

2、lightgbm比较重要的参数是什么
树的深度，叶子节点数，节点分裂的最小增益值，叶子节点的最小样本树，L1正则化系数，L2正则化系数，列抽样比例，训练的轮次

3、不结合业务怎么选择合理的特征
特征选择的方法大体可以分为三类：
(1) Filter
① 按照特征和响应变量之间相关性进行排序，选择相关性最高的部分特征；
② 用每个特征对响应变量进行预测，依评价指标的好坏进行排序，选择评价指标较高的部分特征；
③ 低方差过滤，计算每个特征的方差，如果一个特征的方差特别低，一般认为这个特征的贡献不大，可以剔除；
④ 缺失数据过滤，如果一个特征的缺失率非常高（比如达到90%），那么可以直接剔除；
.....

经典的方法如简建立多个一元回归方程进行评价，使用χ^2统计量进行选择等。

(2) Wrapper
构造一系列的特征子集，然后确定一个评价指标，用这一系列的特征分别建立模型去拟合响应变量，取评价指标的值最高的模型对应的特征子集作为选择结果。

典型的方法如回归分析中的向前特征选择、向后特征选择，逐步回归等。

(3) Embedded
即在建立模型的时候，同步进行特征选择，当模型训练完成时，特征选择的过程也完成。

典型的方法如：Lasso，决策树等

4、在业务中是怎么处理类别型特征的
label encoder，one-hot encoder

牧原（数据挖掘分析师-校招）

1、你了解或者常用的机器学习算法有哪些？

2、GBDT、XGBoost、Lightgbm分别有什么特点？他们之间的关系是什么样的？你对他们有什么认识？

3、你认为一个完整的项目流程应该是什么样子的？
拆解需求 $\rightarrow$ 确定所需数据 $\rightarrow$ 数据清洗 $\rightarrow$ 特征工程 $\rightarrow$ 构建标签/选择响应变量 $\rightarrow$ 选择模型 $\rightarrow$ 模型训练和调优 $\rightarrow$ 重复特征工程和模型训练的过程进行迭代优化 $\rightarrow$ 前后端联调 $\rightarrow$ 模型部署 $\rightarrow$ A/B test

4、怎么处理数据不平衡问题？

5、怎么处理过拟合问题？

6、知识图谱怎么做子图谱的融合的？

7、介绍EGES算法的原理。

8、EGES算法有什么不足？
不能考虑节点类型和关系类型

9、GCN和GNN有什么差异？

10、知识图谱的可落地场景？
u2u召回、u2i召回、i2i召回、用户扩散、知识问答、广告投放等

注：一面为群面，两个面试官为综合面试和技术面试的面试官，和投递同一个岗位的竞争者一起面试！总耗时约一个半小时。二面面试官由HR和技术面试官一起面，总计一小时左右。

海信（校招）

1、uplift模型是怎么发现敏感人群的？
参考营销敏感四象限人群

2、s-learner、t-learner、x-learner之间有什么区别？
s-learner 只训练一个模型，t-learner对treatment组和control组分别训练一个模型

3、怎么评估uplift模型给出的结果？
Qini AUC（AUUC），uplift柱状图，cuplift

4、在因果推断的项目中使用了哪些工具？工程的具体实现分为哪几个部分？
工具：主要使用HiveSQL，Spark，Causalml

工程：选择混淆变量、数据清洗、确定treatment、划分treatment和control组、模型训练和调优、确定敏感人群、模型验证（与业务侧对齐）

5、随机试验是怎么划分实验组和对照组？
依据一定的相似性计算规则，选择最相似的样本构成一个个的样本对，然后将推断变量值高的放入treatment组，推断变量值低的放入control组。

6、划分实验组、对照组时，尝试过哪些相似性指标？最终选择了哪一个？
欧氏距离，余弦相似度，马氏距离，曼哈顿距离

7、怎么确定的使用uplift模型？

歌尔（校招）

1、pr和roc的适用范围
pr更关注正样本，当正负样本数量差别不大的时候，使用pr和roc都可以。

2、roc曲线怎么绘制的
使用每一个预测概率作为划分阈值，分别计算一个（fpr，tpr）数据对。

用所有的（fpr，tpr）数据拟合一条曲线，这个曲线就是roc曲线，曲线下的面积就是auc。

3、常用的防止过拟合的方法

4、怎么处理类别不平衡问题

5、常用的激活函数

6、sigmoid函数的缺陷时什么？
容易出现梯度消失

7、决策树模型节点分裂的准则

8、怎么优化lightgbm的性能？

9、xgb和lightgbm的区别？

10、常用的多元统计学习方法？

11、PCA原理

12、k-means原理

薪人薪事

1、metapath怎么处理不同类型节点的

2、怎么在异质图中表示节点的类型

3、uplift的原理和流程

4、uplift值得计算公式

5、怎么区分共现和因果（啤酒尿布案例）

6、特征选择方法

7、判断多重共线性的方法

8、(编程):有向图的pagerank实现

面试问题合集

注：以下问题记不清是来自哪次面试了，含海尔、京东、百度、Aibee、格灵深瞳、中电达通、直引科技、中电金信、金科览智、微芯研究院、智源研究院（30分钟），零犀科技，国双（GridSum，55分钟）等。

1、（编程）topk。

2、ndarray为什么比dataframe的查询效率高？

3、（编程）快速排序。

4、autoML（自动机器学习，19、20年巨火，目前依然有很多公司在用）。

5、因子分解机（MF）/隐语义模型（LFM）。

6、神经网络中是怎么实现矩阵分解、特征交叉的？

7、（编程）二分查找。

8、推荐系统的冷启动策略（用户冷启动、物料冷启动、系统冷启动）。

9、怎么解决欠拟合问题？
① 过采样，增大样本量
② smote方法合成新样本
③ 通过特征交叉、特征组合方式，增加特征数量
④ 集成方法中可以增加基模型数量，增加迭代轮次

10、（编程）快速排序

11、常用的降维方法。
特征选择：
（1）Filter方法：
① 通过χ^2统计量检验；
② 通过方差选择变量；
③ 通过单变量对因变量的拟合效果进行选择（如拟合多个一元回归模型）；
...

（2）Wrapper方法：
① 特征子集法；

② 逐步回归；
③ 向前回归；
④ 向后回归；
⑤ Relif；
· · ·

（3）Embedded方法：
① LASSO；
② Ridge；
③ elastic-net；
④ Decision Tree；
· · ·

特征提取：
① PCA；
② MDS；
③ LPP；
④ LLE；
⑤ t-sne；
· · ·

12、什么情况下适合用ℓ_1正则化，什么时候适合用ℓ_2正则化，什么时候需要同时使用它们？
① ℓ_1正则化：当变量之间不存在相关性或变量之间的相关性比较低时，适合使用ℓ_1正则化；
② ℓ_2正则化：当变量之间存在多重共线性，或存在多个变量同时发挥作用的情况时（即只选入单个特征时，不发挥作用，同时将他们选入时，发挥明显作用），适合使用ℓ_2正则化；
③ 如果不确定变量之间是否存在相关性、是否有多个变量同时发挥作用时可以同时使用ℓ_1和ℓ_2正则化；或，当变量中既有大量相互独立的变量，又有大量相互作用同时发挥作用的变量时，可以同时使用ℓ_1和ℓ_2正则化。

13、ROC曲线的横轴和纵轴分别代表什么？
ROC空间将伪阳性率（FPR）定义为 X 轴，真阳性率（TPR）定义为 Y 轴。

14、异常点检验方法
①3 σ 法则；
②箱线图（中位数 $\pm$ 1.5R之外的数据即为异常点，R为四分位距）；
③dbscan聚类可以识别异常点；
④孤立森林；

第4部分 场景分析

1 公司特征描述

1.1 互联网公司/手机电脑等智能制造公司

整体上看，要进入互联网大厂的难度较大。主要表现在：
① 笔试难度较大，一般给定具体的场景，从场景中抽象出数学问题，然后再进行编码求解，编码难度一般在leetcode的middle难度；
② 面试难度较大，一般针对项目、实习等进行深究，对算法原理追问较细致，往往会问到一些平时注意不到的点，在面试中往往会设置编程题目，要求现场编码，然后在进行代码优化，编程难度惨开leetcode的middle难度题目，一般为leetcode原题。
③ 以22年暑期实习的标准来看，要进入大厂实习，大多首先要求有过大厂实习经历，否则基本上就只能在暑期招聘结尾的时候捡漏。
④ 竞争人数多，越来越多的人涌入互联网，越来越少的岗位被提供，竞争越来越大；以cv岗为例，大多要求有顶会论文，或者博士学历，否则基本只能捡漏。
⑤ 技术要求较多，例如git，sql，spark，scala，java，python，c++等，会的越多，加分越多；机器学习和深度学习均有实际经验的优先入选。

1.2 传统制造业（冰箱、空调、汽车等）

相较于互联网公司，难度较小，体现在：
① 笔试难度小，形式与互联网基本相同，但题目理解难度和编码实现难度较低，难度普遍停留在冒泡排序、斐波那契数列、二分查找的等级上。
② 面试难度一般较小，对算法原理的细节追问较少，对统计学的理论知识的细节要求较多，例如假设检验原理、常见分布的数字特征、ROC-AUC的计算原理等。
③ 面试中一般不设置编程。
④ 技术要求不多，但会的越多，加分越多。

1.3 银行

经验不多，待补充

1.4 个人体验感

先后使用过快手（国内互联网）、斯伦贝谢（外企、石油）、腾讯体验卡。

快手：规定工作时间早10晚7，实际工作时间下不封底，晚上8点钟几乎没人离开，9点钟大有人在，十点钟开始企业滴滴打车；整体工作强度大，日常除了上厕所没有休息时间，午休2小时，除去午餐时间，实际只有半个多小时休息，晚上基本就是吃完饭就开工。领导精神pua，阿里文化，每天日报，精神、身体双重破防，50天果断离职。
薪资方面：日薪450，30餐补券（加班到20：00以后打卡发放），周六周末加班的话，每4小时一张餐补券，每天最多两张，没有房补。

斯伦贝谢：规定工作时间早9晚6，实际上9点半上班，16：30就可以离开了。待了22天，全部在学习leader发的学习材料，感觉比较枯燥、空闲，所以离职了。
薪资方面：全勤每月6000，早上9：30之前到公司能吃到免费早餐，午、晚餐自费，无房补。

腾讯：规定工作时间是早10晚7，实际十点左右到公司，18：00左右就可以去打饭回家了，实习生没有企业滴滴权限，公司不鼓励实习生加班，午休两个小时，晚餐后基本没人回去加班。平时工作强度也不是很大，自我感觉是互联网公司里面我最能接受的。
薪资方面：实习生全勤的话，7000底薪+2000房补，第一个月有交通补贴（省外学生2000，省内1000），第二个月有3000加油包，早晚餐免费，午餐自费，实习生不享受企业滴滴，但是如果因为工作必须要加班，可以把发票给认识的正式员工，让正式员工一起给报销（公司条例就是这么说的）。

2 方法总结

整个流程一般为：准备简历--投递简历--综合测试（暑期实习和校招会有，日常实习可能没有）--笔试（日常实习可能没有）--面试（大厂暑期实习一般三轮面，小厂也有一轮的；非互联网一般为一轮面）--hr面--喜提offer--入职办理

2.1 简历整理

简历筛选是第一步，因此一定要好好准备简历。
① 简历格式要简约，尽量不要是那种花花绿绿带各种颜色的，不要是那种棱棱角角带有很多图形元素的，很影响第一感觉。
② 简历内容要简明扼要，元素一般包含：姓名，出生年月，意向工作地，意向岗位，联系方式（电话、邮箱、git），证件照，毕业院校及专业（本、硕），研究方向，主要课程，实习/项目/

竞赛经历（可附加超链接），技术栈（编程语言、使用工具、理论框架等），任职（校内、外，本、硕）、荣誉奖励等。
③ **建议将实习、项目、竞赛等成果整理成文件，上传至git等可以接受他人访问的网站或个人主页，然后将成果链接附到简历中，方便面试官和hr查看。**
④ 实习、项目、竞赛等的介绍要简洁、突出成绩和结果，最好用一些数字来体现，例如，开发了××算法，AUC为96.6%，全量用户的A/B测试中，日活用户提升了6.6个pp。

2.2 简历投递

很多同学的研究方向和当前互联网的职位需求并不相似，就觉得自己不适合相关的岗位，陷入认知误区。一般来说，职位之间存在一种兼容关系，即算法兼容数据开发，数据开发兼容数据分析，数据分析兼容产品运营。此外以算法岗为例，传统机器学习算法岗和深度学习的算法岗存在较大差异，但也不是不能投、不能做，因为实习本来就是来丰富自己的能力的，就是想做一些没机会使用的技术，而且很多公司都会给这样的机会，所以，只要自己有兴趣就可以投。

投递简历的几种方式：公司官网一般会提供暑期实习、日常实习、校招、社招几种招聘入口；还可以通过BOSS直聘（可以通过某宝购买账号）、实习僧、智联招聘、拉钩等APP投递简历（依效率高低进行的排序）。

2.3 综合测试+笔试

强调一点，在投递简历、笔试、面试的任何一个环节都很重要，不要忽视。

综合测试
综合测试都是要做的，否则不会安排笔试和面试。
综合测试不可以随意回答，会影响是否能进入笔试和面试。
我理解为，综合测试主要考察是否有性格缺陷，情绪偏激，能否于人正常相处，回答是否前后一致，能否理解文字内容表达的意思，能否都懂图表所表达的意思等。

笔试
笔试是简历筛选通过后的第一个难关，一般会使用牛客或者赛码，可以先体验一下两个平台的使用方式，代码模式：核心代码模式（参考leetcode），ACM模式（注重输入输出的设计）。
牛客需要扫码小程序，即手机电脑两台设备都不能作弊。
赛码可以使用手机搜题。

2.4 面试

面试的第一个环节就是自我介绍，所以提前准备一套比较全面、简洁的自我介绍也很重要，大致就是把简历的内容捋一遍。然后面试官一般会针对你的项目/实习/竞赛进行发问，因此要对自己写的内容有足够的了解，比如你在简历上写了某个竞赛，用了哪些算法，那就要知道你用的这些算法的原理，实现方法，内部的一些处理逻辑，怎么优化，他的特点是什么，相对于其他方法有什么优缺点，做过哪些改进，为什么做这些改进等等。互联网公司在最后一般会安排一道编程题，如果不是在他们自己的面试系统上，可以调出自己以前写过的代码，如果是在他们自己的面试系统上，可以看机会偷摸手机搜一下题。

一些注意点：比如你投的岗位和自己的方向不符合的时候，面试官可能会问，为什么投一个自己不了解的岗位？我之前都是回答，因为导师研究方向的原因，没有机会接触这个领域，但是自己对这个也非常感兴趣，向在实习中学习一下这方面的内容，拓展一下自己的技能，一般中小公司会给过的（亲测，拿过几个不相关方向的实习offer），大厂没有实际经验。

传统企业会问一些类似于，家在哪里，想在哪里发展，意向城市有哪些等等的话题，其实就是想看你能不能稳定的在这个公司发展，他们不希望说花了几年时间培养了一些人，然后都跑了。所以即使仅仅是为了实习，也要顺着他们的意思回答，至于实习完了留不留，这个选择权还是在自己手上，传统企业实习完是有很大概率能转正的。

面试官在最后一轮面试的时候往往会问一下期望薪资，这个可以提前参考offershow微信小程序，按照以往的薪酬略微提高一些；此外，不要说期望薪资的区间，因为他们只会给区间下限，一定按照期望薪资的上限说一个具体值。

3 其他补充

有些公司是电话面试，例如格灵深瞳、比亚迪。

有些公司的综合面试是群面，例如美的、三一；
美的群面形式：每人一分钟自我介绍，然后五分钟读题（例如给定单身青年人群，智能扫地/拖地机器人产品，讨论其市场需求和痛点，给出一些解决方法、设计方案），然后二十分钟的讨论环节，最后是推一个人进行五分钟的最终总结。
三一群面形式：每人100秒自我介绍，然后面试官逐个提问，大概问题就是，你为什么选择三一，家在哪里，为什么不选择互联网。
牧原群面形式：两个面试官（HR和技术面试官），两个应试者