数据结构相关问题：

哈希冲突的解决方法：

开放定址法：冲突时探测下一个空槽（如线性探测、平方探测）。

链地址法：冲突位置用链表存储所有冲突元素。

再哈希法：使用第二个哈希函数计算新位置。

基础问题：

为啥二分类问题里面会用到交叉熵损失，而不用平方损失呢？

Sigmoid 会出现梯度消失，mse 的梯度更新中有导数，会导致梯度无法更新，而交叉熵的导数就直接是误差，不会导致梯度无法更新

Mse 衡量的是数值距离，而交叉熵衡量的是真实分布与预测分布的差异

交叉熵的梯度更新更快，直接与误差有关

ROC 曲线和 AUC 值的意义是什么？如何根据 AUC 评估模型性能？

ROC 曲线衡量二分类模型在不同分类阈值下的假阳率和真阳率（召回率）

AUC 值表示 ROC 曲线的面积，AUC 越大表示模型性能越好

监督学习与无监督学习的核心区别是什么？列举两者的典型算法。

输入数据有无标签

监督学习—常见的语言模型的微调都是监督学习，无监督学习就比如说 k-means

什么是过拟合？如何解决过拟合问题？

在训练集上表现很好，但是在测试集上表现很差

正则化

交叉验证

增大数据量

数据增强

交叉验证的作用是什么？K 折交叉验证的具体流程如何实现？

可以更充分地利用数据。将数据分为 k 折，迭代训练 k 次，每次取其中 k – 1 折数据为训练集，剩下 1 折为验证集，

机器学习相关问题：

介绍 xgb、lgb、cat：

这三个都是类似的都是 GBDT（梯度提升树），用 boosting 的方法迭代训练多个弱决策树后集成。

XGB 是在 GBDT 的基础上优化损失函数。加入正则化控制树的复杂度。相比 GBDT 使

用一阶导，XGB 使用二阶偏导近似提高准确度。正则化的变量包括叶子节点的数量和叶子节点值（值小一些树可以多一些）。

二阶偏导优化后可以得到第 k 棵树不同形状每个节点最优的权重值，用层序的贪心算法生成新节点的方式，比较最有权重值下的 loss 来选取树的形状。

LightGBM 是在 XGB 基础上改进了，直方图决策树和将构建树形状的方法改为用 leaf-wise 的贪心算法。直方图将特征从连续变为离散，减少计算次数。Leaf-wise 每次从所有叶子节点中找到分裂增益最大的进行分裂，而非必须层序生成新节点。

CatBoost 能够对特征进行处理 one-hot 或者其他的特征编码，并且增加组合特征来提升特征融合能力，采用对称决策树

总结一下就是，XGB LGB CAT 都是 GBDT，LGB 和 CAT 是在 XGB 的基础上的改进，XGB 是对传统 GBDT 的改进，用 boost 的方法迭代训练 k 棵决策树然后集成，XGB 采用 level-wise 的贪心算法生成树的形状，并且加入了包含叶节点数量和叶节点权值的复杂度正则化。LGB 在 XGB 的基础上用直方图的方法改进了特征划分的方式，减少了特征的划分数量，并采用有深度限制的 leaf-wise 的贪心算法来生成决策树。CAT 就是在特征上采用自动特征编码，并增加组合特征来提高对特征的理解，并且将生成树的形状要求为对称决策树避免过拟合。

比赛中把语言模型的预测结果和 XGB LGB CAT 集成的原因是:

语言模型更适合理解上下文的关系，但是对结构化特征数据和有规则的数据理解能力没有机器学习模型强，所以集成机器学习模型

你刚刚提到了 boosting 的方法，那请问另一种集成学习的方法，bagging 是什么?

Bagging 是对数据集进行有放回抽样，然后对每个子数据集训练一个模型，最后对这些数据集进行软投票或硬投票。这些子数据集其实有时候也可以是 K-fold。

介绍一下 svm 支持向量机

Svm 是要求找出一个超平面能够将不同的分类分开，并最大化两类数据之间的间隔

介绍一下 k-means

k-means 是选择 k 个数据作为中心点，然后将其他数据以最短欧几里得距离分配到每个点，然后得到 k 个簇，求每个簇内的均值，并将其作为新的中心点。然后不断循环。

BGE GTE BM25 TF-ITF BGE-reranker 是什么

BGE 和 GTE 是中文 embedding 模型，把文本映射到高维向量，然后相乘计算和查询问题的相似度，用于粗排

BGE-reranker 是端到端的捕捉细粒度关联的，对细节方面效果更好，所以用于精排

TF-IDF 是词频逆文档分数，反映查询词与文档的相关性

2. TF-IDF 的计算公式

TF-IDF 的计算公式如下：

(1) 词频（TF）

词频表示一个词在文档中出现的频率。常见的计算方法有两种：

- **原始词频**：

$$\text{TF}(t,d) = \frac{\text{词 } t \text{ 在文档 } d \text{ 中出现的次数}}{\text{文档 } d \text{ 中的总词数}}$$

- **对数词频**：

$$\text{TF}(t,d) = \log(1 + \text{词 } t \text{ 在文档 } d \text{ 中出现的次数})$$

(2) 逆文档频率（IDF）

逆文档频率表示一个词在文档集合中的稀有程度。计算公式为：

$$\text{IDF}(t,D) = \log\left(\frac{\text{文档集合 } D \text{ 中的总文档数}}{\text{包含词 } t \text{ 的文档数}}\right)$$

- 如果一个词在所有文档中都出现，其 IDF 值为 0。
- 如果一个词在很少的文档中出现，其 IDF 值较高。

(3) TF-IDF

TF-IDF 是 TF 和 IDF 的乘积：

$$\text{TF-IDF}(t,d,D) = \text{TF}(t,d) \times \text{IDF}(t,D)$$

BM-25 是 TF-IDF 的变体

TF 项增加词频饱和度、文档长度的影响

IDF 项引入平滑项

介绍一下 JIEBA 分词?

Jieba 用来做中文分词的库，两个比赛都用的 jieba 默认的精确分割模式，比较适合分割关键词。

为什么文档检索需要分两个不同的块?

Pdf 分为长的和短的分块，长的保留上下文语义信息，短的可以捕捉更多关键细节信息。

深度学习比赛相关问题

你的比赛中使用了 RoBerta，它和 bert 有什么提升吗?

Roberta 使用了更大的 batch_size，预训练 roberta 时，使用动态掩码训练过程中实时生成掩码，而 bert 是静态掩码，数据预处理时就生成掩码。Roberta 不使用 next-token-prediction 只使用 masked language modeling，bert 是都是用，但实际上 bert 类模型只使用 masked language modeling 更好。

OCR 使用的什么?

使用的是 PaddleOCR

介绍一下 clip

Conv2d:

```
nn.Conv2d(
    in_chans,        # 输入通道数
    embed_dim,       # 输出通道数
    kernel_size,     # 卷积核的**长和宽**（如 16×16）
    stride           # 卷积核移动的**步长**（如 16）
)
```

有 embed_dim 个卷积核，卷积核的长宽为 kenel_size，移动步长为 stride，卷积核的高度为 in_chans

用 vit 作为图像编码器，transformers 类语言模型作为文本编码器。

图像从 vit 得到的图像特征与类别经过 prompt 后得到的文本从文本编码器得到的文本特征做矩阵乘法，得到的跨模态矩阵让对角线尽量大，非对角线尽量小，使得文本和图像的输出向量尽可能对齐。
使用方法：
Clip 的图像编码器和文本编码器都是跨模态对齐过的，所以会比直接使用 vit 或者 bert 更能理解图像和文本信息。可以单独将图像和文本分类头拿出来使用。
也可以将文本和图像的跨模态特征拿出来使用

介绍一下 vit：

Vit 是将图像经过卷积分块为多组嵌入特征后，添加 cls_tokens 后补充位置编码，输入 transformers 最后只取 transformers 输出的分类头向量用来分类。
也可以不加 cls_tokens，直接对最后的输出进行平均池化操作，但这样可能会受到无关 patch 的干扰，cls_token 可以看到全局信息。

逻辑回归和 softmax 的区别？

逻辑回归一般用于二分类，他是最后用一个 sigmoid 使得 logits 分布在 0 与 1 之间作为概率，预测时大于 0.5 为正类，小于 0.5 为负类，或者根据概率采样为正或负。
Softmax 一般用于多分类，比如 gpt 流式输出时每个 tokens 的输出分类，是取指数后让所有 logits 的和为 1 即概率总和为 1，最后预测时取概率最大的或者根据概率进行采样。

介绍一下 vllm

他是对注意力 K 和 V 等进行管理，kv-cache 之类的。

然后包含一些量化操作 GPTQ、INT8 等加快推理速度

推理的时候对张量采用流水线并行推理，来加快推理的速度

介绍一下 LoRA

LoRA 是冻结原模型的参数，用线性层去模拟原模型的 qkvo 等，减少微调时的参数量，又能达到不错的效果，LoRA 初始化时 B 初始化为 0 矩阵，A 初始化为高斯分布。A 不初始化为 0 矩阵的原因是 B 根据链式法则梯度更新需要 A 这边传过来，如果时 0 则无法更新了。然后 B 初始化为 0 的目的是，使得初始时基础模型和原模型一样

介绍一下 relu 和 sigmoid 激活函数

Relu 一般用在中间层，可以让梯度下降的更快，并且可以防止梯度消失，sigmoid 进入饱和区时梯度太小

Multi head Attention

QKV 分成 head_dim 组，每组分到 dim / head_dim 个特征维度。

GQA

将 q 分成多组，每组共享同一个 k 和 v，每个 q 分到 dim / group_dim 个维度，每组的 k 和 v 要广播和 q 计算。

Qwen2.5-instruct-GPTQ-Int8 的创新点：

GQA

GPTQ 量化压缩精度

离线 DPO

更大的训练数据集

INT8 量化

把浮点数 FP32 转为 8 位整数，先确定缩放的比例，然后将浮点数缩放后四舍五入为整数存储，然后反量化就是除以缩放的比例得到 FP32

DPO

构建偏好数据集和非偏好数据集，然后令损失函数最大化偏好数据的概率，最小化非偏好数据的概率。并且引入参考模型，将参考模型对数据的偏好引入损失，使得需要改进的模型不会偏离参考模型太多。

Tokenizer

bpe 用拆分之后根据词频不断合并的方法构建词表和合并策略

Deepseek 相关问题：

Multi head latent attention

是把输入 X 先经过线性层映射为 Z，再把 Z 经过 qkv 矩阵得到 QKV

其他的 attention 是直接把 X 经过 qkv 矩阵得到 qkv

Moe

有两个组件，一个门控组件，一个专家组件

门控组件决定每个数据由 score 排前 top_k 的专家输出，数据输入对应的专家后，得到的输出进行平均得到结果，如果有共享专家，则加上共享专家的结果。

PPO

$$Loss_{ppo} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} A_{\theta'}^{GAE}(s_n^t, a_n^t)\frac{P_\theta(a_n^t|s_n^t)}{P_{\theta'}(a_n^t|s_n^t)} + \beta KL(P_\theta, P_{\theta'})$$

$$Loss_{ppo2} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} min(A_{\theta'}^{GAE}(s_n^t, a_n^t)\frac{P_\theta(a_n^t|s_n^t)}{P_{\theta'}(a_n^t|s_n^t)}, clip(\frac{P_\theta(a_n^t|s_n^t)}{P_{\theta'}(a_n^t|s_n^t)}, 1-\varepsilon, 1+\varepsilon)A_{\theta'}^{GAE}(s_n^t, a_n^t))$$

利用学习参考模型，实现 off-policy

根据 reward 衰减和训练价值函数用来估计优势函数。

价值函数需要预测当前的 reward 衰减和

设置 KL 散度防止偏离参考模型太多，或采用 clip 在偏离很多的情况下进行裁剪。

当参考模型的优势函数大于 0 时使得当前的动作概率尽可能大，小于 0 则使得当前动作概率尽可能小

GRPO

2. GRPO的损失函数

GRPO省略Critic网络，优势函数由组内归一化奖励替代：

$$L^{\text{GRPO}} = \underbrace{\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{\|o_i\|}\sum_{t=1}^{\|o_i\|}\min\left(r(\theta)\hat{A}_i^t, \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i^t\right)\right]}_{\text{组内裁剪损失}} - \underbrace{\lambda\mathbb{E}\left[\text{KL}(\pi_\theta\|\pi_{\text{ref}})\right]}_{\text{KL散度惩罚}}$$

- **$\hat{A}_i^t = \frac{r_i - \mu_{\text{group}}}{\sigma_{\text{group}}}$**：组内归一化优势，$\mu_{\text{group}}$和$\sigma_{\text{group}}$为组内奖励的均值和标准差 [1] [7]；
- **G**：组内候选样本数量（如5个答案） [6]。

在 PPO 的基础上省略了价值函数的训练，每次生成多个候选预测作为候选预测组，然后根据组内的标准差和均值标准化奖励，使得优势值能够体现不同预测在不同环境下的相对优势和相对劣势。PPO 其他部分都保留。相比 PPO 能够节省更多的显存。组内对比也可以减少迭代次数加速收敛。也可以避免因为价值函数的偏差导致的策略振荡。

高质量思维链（Chain-of-Thought, CoT）

DeepSeek-R1 在 zero 的基础上用 CoT 来分多步骤微调，先微调输出格式，然后再微调 answer 部分，使得其在数理逻辑和代码能力上表现很好