同济大学

实习日记本

学　　院	计算机科学与技术学院
专　　业	数据科学与大数据技术
学　　号	2251764
学生姓名	江来
接受单位	和鲸社区
地　　址	线上
指导教师姓名	郭玉臣

大学生校外实习守则

一、高校校外实习是高等教育同社会主义建设相结合的重要环节，是培养合格人才的重要保证，每个学生必须按照教学计划的要求，认真参加各种形式的校外实习活动，并自觉地遵守本守则。

二、严格遵守实习单位的有关规章制度。如有违反，学校应会同实习单位按有关规定严肃处理。

三、自觉遵守劳动纪律、尊重实习带教人员，服从安排，严格执行安全操作规程。

四、妥善保管因实习需要而借阅的图纸、资料，严格遵守保密制度，放置泄密。

五、认真写好实习日记，客观记录实习的内容、问题及心得。

六、独立完成实习报告，系统总结、综合分析实习中的问题以及解决问题的方法和建议。

七、实习结束时，必须如数归还所借用的劳防用品、工具、仪器、资料及生活用品，如有损坏或遗失应按规定赔偿。

八、学生在实习期间造成伤残和死亡的，可按“大学生团体平安保险”有关规定给予赔偿。

实习起迄日期
自 2025 年 7 月 13 日起至 2025 年 8 月 12 日止
实 习 内 容
本次实习是同济大学计算机系的集中实习，我选择了和鲸社区作为实习单位。实习采用线上的方式，目标是完成和鲸社区暑期夏令营活动。这段实习历时一个月，涵盖基础知识、机器学习、深度学习、自然语言处理（NLP）、模型微调、蒸馏与 RAG 等多个领域，从理论学习到实战操作，层层深入，让我对人工智能技术的应用有了系统且深刻的认识。 实习初期，我从 python 基础入门开始，每天认真完成任务，逐步回忆起 python 语法，之后的 numpy、pandas、matplotlib、seaborn 等常见库的学习为后续深入学习打下基础。中期进入机器学习和深度学习项目，端到端的理论基础和实战经验加深了我对机器学习和深度学习工程落地的认知。 实习后期的三个高难度实战任务给了我极大挑战。LoRA、知识蒸馏、RAG 实战，从数据处理到检索与生成模块的搭建，每一项都面临着诸多问题和重重挑战，虽未达到理想效果，但让我认识到实际落地时细节打磨的重要性。 总的来说，这次和鲸社区暑期夏令营实习虽然充满挑战，甚至不少任务以失败告终，但每一次尝试都让我收获颇丰。我不仅掌握了多种算法和模型的原理与应用，更在实战中积累了处理复杂问题的经验，深刻理解了理论与实践之间的差距。这段经历为我今后在计算机领域的深入学习和研究奠定了坚实基础，也让我对人工智能技术的落地应用有了更清晰的认知。
指导教师批阅意见： 指导教师签名　　月　日

7 月 13 日 第一天 星期日

今天是参加“和鲸社区联合同济大学暑期实习”活动的第一天，主题是人工智能与大模型开发。虽然我对 Python 已经有了一定的基础，但今天的任务是从最基础的 Python 语法开始，系统地回顾和巩固知识。尽管内容相对简单，但温故而知新，这个过程让我对 Python 的理解更加清晰和扎实。

首先我完成了关卡 1 的学习。这一部分主要介绍了 Python 的基本特点、以及运行环境等。虽然这些内容对我来说已经非常熟悉，但我还是认真阅读了材料，并按照要求完成了简单的“Hello World”程序。通过这一关卡，我重新梳理了 Python 的发展历史和它在数据分析、人工智能领域的广泛应用。

接着是关卡 2 的学习。这一部分主要讲解了 Python 的变量命名规则、注释的使用、代码缩进的重要性以及基本的输入输出操作。虽然这些知识点我已经熟练掌握，但通过实际操作，我进一步巩固了对 Python 语法规则的理解。特别是在代码缩进的部分，我重新认识到 Python 对代码格式的严格要求，这与其他编程语言（如 C++）有很大的不同，也是 Python 简洁易读的关键之一。

今天的实习虽然内容基础，但让我意识到系统化学习的重要性。即使是熟悉的知识点，通过结构化复习也能发现新的细节和思考角度。

7 月 14 日 第二天 星期一

今天是实习的第二天，继续深入学习 Python 的基础知识，重点涵盖了数据结构、控制流和函数三大核心内容。虽然这些概念我之前已经掌握，但通过系统的复习和实践，我还是发现了一些之前忽略的细节。

在关卡 3 中，我系统回顾了 Python 的常用数据结构，包括列表、元组、集合、字典等。虽然这些数据类型的基本操作对我来说并不陌生，但通过练习，我更加清晰地总结了它们的特性和适用场景：

- 列表的灵活可变性使其适合存储动态数据；
- 元组的不可变性保证了数据的安全性；
- 集合的无序性和唯一性在去重和集合运算中非常高效；
- 字典的键值对结构在快速查找和映射关系时极具优势。

我还尝试了一些进阶操作，比如列表推导式和字典推导式，这些简洁的语法让代码更加优雅高效。

在关卡 4 中，我复习了 if 和 for 的使用。虽然这些逻辑结构很基础，但它们是构建复杂程序的基石。我通过几个小练习巩固了 break、continue 等控制语句的用法，并尝试用循环优化了一些数据处理逻辑。

关卡 5 则重点讲解了函数的定义与调用、参数传递、返回值等。。此外，我还尝试了 lambda 表达式，lambda 表达式的灵活运用也让我印象深刻，它让简单的匿名函数编写变得更加便捷。

今天的实习让我对 Python 的基础知识有了更系统的梳理，虽然内容不算新，但还算有所收获。

7 月 15 日 第三天 星期二

今天正式开始了 NumPy 模块的学习，这是 Python 科学计算的基础库，在机器学习和深度学习领域有着广泛应用。虽然之前对 NumPy 有一定了解，但通过今天的系统学习，我对它的核心功能有了更深入的认识。

首先是数组的创建与基本运算，我复习了 ndarray 的各种创建方式，包括从列表转换、使用特定函数生成等。向量化运算的特性让我再次感受到 NumPy 相比原生 Python 列表的效率优势。第二关讲解了数组的聚合操作和广播机制，这些功能在数据处理时非常实用。最后学习了数组的索引和比较，特别是布尔索引和花式索引的使用技巧，这对后续的数据筛选和预处理很有帮助。

通过今天的实践练习，我更加熟悉了 NumPy 的高效计算方式。比如在处理矩阵运算时，合理利用广播机制可以避免不必要的循环；使用聚合函数能快速计算统计量；而灵活的索引方式则让数据选取变得轻松。这些技能都是进行科学计算和数据分析的基础。

虽然内容不算新颖，但系统性的复习让我发现了一些之前忽略的细节，比如不同创建方式的性能差异、广播机制的具体规则等。这为后续更复杂的数值计算和机器学习算法实现打下了坚实基础。期待明天继续深入学习相关的内容。

7 月 16 日 第四天 星期三

今天继续深入学习 NumPy 的进阶功能，并完成了综合实战项目，收获颇丰。今天主要完成后面两个关卡：数组排序与结构化数组的应用，以及会员消费与积分分析的实战演练。

第一关我重点掌握了 NumPy 的排序算法。通过实践，我熟悉了排序函数的应用技巧。结构化数组部分让我对处理复杂数据类型有了新的认识，特别是在处理带有不同数据类型的表格数据时，结构化数组展现出了独特的优势。

接下来的综合实战项目"会员消费与积分分析"让我将所学知识应用到实际场景中。通过分析会员消费数据，我练习了数据清洗、特征计算、统计分析等完整的数据处理流程。这个实战项目不仅巩固了 NumPy 的各项操作，还让我体会到数据预处理在真实业务场景中的重要性。在解决具体问题时，我特别注意了代码的效率和可读性，这对培养良好的编程习惯很有帮助。

通过今天的学习，我对 NumPy 的理解更加深入，特别是在处理实际业务数据时，能够更合理地选择适当的方法和函数。期待明天开始新的学习内容，继续提升数据处理能力。

7 月 17 日 第五天 星期四

今天正式开始了 Pandas 库部分的实习，这是 Python 数据、文件分析的核心工具。通过今天的实习，我对 DataFrame 这一核心数据结构有了系统的认识，掌握了基本的数据处理技能。

第一关中我重点学习了数据的读取与存储，包括从 CSV、Excel 等常见格式读取数据的方法，以及如何将处理结果保存到文件中。通过实践，我熟悉了 DataFrame 的基本属性和常用操作，如查看数据概览、处理缺失值等。这些基础操作看似简单，但却是后续复杂数据分析的重要基础。

第二关深入讲解了 Pandas 的索引系统，包括位置索引、标签索引以及布尔索引等多种数据选取方式。特别是多层索引(MultiIndex)的使用让我印象深刻，这种灵活的索引方式可以很好地处理高维数据。这一部分我之前学的并不是很好，这次通过实际练习，我掌握了 loc 和 iloc 等索引方法的区别与适用场景，这对后续的数据筛选和切片操作很有帮助。

今天的实习让我认识到，虽然 Pandas 的基本操作不难掌握，但要真正熟练运用还真是需要大量的实践。特别是在处理真实业务数据时，合理的索引选择能显著提高代码效率和可读性。期待明天继续深入学习 Pandas 更强大的数据处理功能。

7 月 18 日 第六天 星期五

今天深入学习了 Pandas 的数据清洗核心操作，并重点掌握了表连接(merge)这一关键技能。通过将 Pandas 操作与 SQL 进行对比，我对数据处理有了更系统的理解。

在数据清洗方面，我系统练习了数据的新增(concat/append)、删除(drop)、查询(loc/iloc)和修改(assign)等核心操作。这些操作让我联想到 SQL 中的对应语句，但 Pandas 的链式调用方式让代码更加简洁流畅。特别是在处理缺失值时，Pandas 的 fillna()和 interpolate()方法比 SQL 提供了更丰富的填充选项。

今天的重点收获是深入理解了表连接操作。通过 merge()函数，我实践了内连接(inner)、左连接(left)、右连接(right)和外连接(outer)等不同连接方式，这与 SQL 中的 JOIN 操作完全对应。我特别注意到了 on、left_on/right_on 参数的使用场景差异，以及 how 参数对连接结果的影响。例如在处理用户订单数据时，左连接可以保留所有用户记录，即使他们没有下单记录。

通过实际案例，我发现 Pandas 的 merge 在功能上比 SQL 更加灵活：

1. 支持更复杂的多列连接条件
2. 可以方便地控制结果表的列名
3. 能够直接处理 DataFrame 而不需要建立数据库连接

今天的实战实习让我认识到，表连接是数据分析中最常用的操作之一，掌握各种连接方式的适用场景对保证数据准确性至关重要。Pandas 在这方面的表现确实令人印象深刻，期待明天继续探索更多高级功能。

7 月 19 日 第七天 星期六

今天开始学习 Python 数据可视化的重要工具 Matplotlib，通过实践掌握了基本图表的绘制方法和图表元素的配置技巧。

在基础图表部分，我重点练习了六种常见图形的绘制方法。柱形图和条形图适用于类别对比，折线图擅长展示趋势变化，面积图能直观呈现累积效果，饼图适合展示比例关系，而散点图则能揭示变量间的相关性。通过实际案例，我体会到不同图表类型的选择对数据表达的重要性，比如在展示月度销售额时，折线图比柱形图更能突出变化趋势。

图表元素配置环节让我收获颇丰。我学会了如何调整画布尺寸来适应不同展示场景，解决了中文显示乱码的问题，并掌握了添加标题、坐标轴标签、图例等关键元素的方法。特别是通过设置数据标签和网格线，大幅提升了图表的可读性。在调整坐标轴范围时，我发现合理的范围设置能让数据特征更加突出。最后，通过修改图标风格，我制作出了更专业美观的图表。

今天的实践让我认识到，优秀的数据可视化不仅需要准确传达信息，还要注重细节处理。Matplotlib 虽然学习曲线较陡，但其灵活性和强大功能令人印象深刻。

7 月 22 日 第八天 星期二

今天深入学习了 Matplotlib 的复杂场景应用，并开启了 Seaborn 的入门之旅，两种工具的特性与用法差异给我留下深刻印象。

Matplotlib 的进阶操作充满挑战。多图布局中，subplots 函数的行列参数设置需要精准把控，通过反复调试子图的矩阵排列，我终于理清了参数逻辑。双 Y 轴图表的绘制关键在于轴的绑定与视觉区分，左侧蓝色折线与右侧红色虚线的搭配，让两组差异较大的数据既清晰分离又相互关联。fill_between 函数的阴影效果能强化数据区间特征，alpha 值 0.2 的设置既不抢眼又能凸显层次；annotate 标记则需细致调整箭头角度，避免遮挡关键数据点。

而 Seaborn 的简洁高效令人惊喜。一行 sns.lineplot () 代码就能完成 Matplotlib 需多步配置的美化工作，自动优化的坐标轴与网格让图表瞬间专业起来。其颜色配置体系尤为出色，cubehelix 色表的渐变效果比手动调配 RGB 更自然和谐，能直观增强数据的区分度。不过图表层级中 legend 与 hue 参数的联动逻辑较为复杂，需要结合案例反复琢磨才能灵活运用。

今天的学习让我明白，数据可视化工具的选择需结合场景需求：Matplotlib 的高度自定义适合精细化设计，Seaborn 的封装特性则适合快速产出高质量图表。明天计划用 Seaborn 重构旧图表，检验今日所学的实际应用效果。

7 月 23 日 第九天 星期三

今日继续深入学习 Seaborn，重点学习了基本图表类型与组合图表的构建方法，对其数据可视化逻辑有了更系统的认识。

基本图表部分覆盖了五类核心应用场景。相关性图表中，heatmap 函数通过颜色深浅直观呈现变量矩阵的相关系数，annot=True 参数让数值标签与色块形成呼应，轻松定位强相关因子；分布型图表里，displot 的 kde 参数能自动生成核密度曲线，比单纯的直方图更能揭示数据分布形态。分类型数据可视化时，boxplot 与 violinplot 的组合使用很有讲究，前者清晰展示四分位线，后者补充分布密度特征，两者结合让类别差异分析更全面。回归模型图表中，regplot 的 ci 参数可控制置信区间的显示，帮助判断拟合结果的可靠性。

组合图表的学习让数据呈现维度更丰富。FacetGrid 通过 row/col 参数实现数据的多维度拆分，绘制出按类别分组的子图矩阵，尤其适合多变量交叉分析；PairPlot 自动生成变量间的散点图与分布图矩阵，能快速发现数据间的潜在关联，但需注意样本量较大时的运行效率问题。JointPlot 则巧妙融合了散点图与直方图，主副图联动展示双变量关系与单变量分布，用 kind 参数切换 “reg” 模式还能直接添加回归线。

经过一天的实践发现，Seaborn 的图表设计始终围绕 “数据关系” 展开，无论是单一图表还是组合形式，都在引导使用者更高效地挖掘数据规律。

7 月 24 日 第十天 星期四

今日正式开启机器学习入门之旅，因为这部分比较难，我也想重新系统地学习一次，所以每天做一个关卡，争取完全弄懂。首个关卡聚焦逻辑回归模型，通过乳腺癌分类与二手车售价预测两个案例，深入理解了这种经典模型在分类与回归任务中的应用逻辑。

乳腺癌诊断案例让我吃透了分类问题的核心思路。首先明确任务是通过肿瘤的 10 个特征（如半径、纹理、光滑度等）的均值、标准差和最大值，判断肿瘤为良性或恶性。逻辑回归通过 sigmoid 函数将线性组合结果映射到 0-1 区间，以此输出患病概率。实践中，我掌握了数据划分的关键：用 train_test_split 按 8:2 比例拆分训练集与测试集，通过交叉验证调整正则化参数 C，平衡模型复杂度与泛化能力。评估时发现，单纯看准确率不够全面，在医疗场景中，召回率（避免漏诊）与精确率（减少误诊）需协同考量，而 ROC 曲线下的 AUC 值则更直观反映模型整体性能。

二手车售价预测则展现了逻辑回归在回归任务中的变体应用。与分类不同，这里模型直接输出连续的价格预测值，损失函数采用均方误差。特征处理是关键环节：对车龄、里程等数值特征做标准化，对品牌、燃油类型等类别特征用独热编码转换，避免引入不合理的数值关系。通过对比不同特征组合的 R^2 评分发现，车龄、里程和发动机功率对售价影响最显著。

今天的学习让我明白，逻辑回归的核心是建立特征与目标变量的线性关系，但其灵活性能适应不同任务场景。区分分类与回归的本质在于目标变量的类型，而特征工程和评估指标的选择则直接决定模型的实用价值。

7 月 25 日 第十一天 星期五

今日聚焦 KMeans 聚类算法，以股票数据为样本实操分类任务，深刻体会到无监督学习在数据探索中的独特价值。与有监督学习依赖标签不同，KMeans 通过挖掘数据内在特征自动分组，这种 “让数据自己归类” 的逻辑为股票分析提供了新视角。

本次案例使用的股票数据集包含丰富维度：核心特征涵盖交易相关的 trade、pricechange、changepercent 等共 19 项数据。

特征选择方面，考虑到估值与流动性是股票分类的核心维度，最终选取 per（市盈率）、pb（市净率）、changepercent（涨跌幅）、turnoverratio（换手率）四项特征。预处理时，发现 per 与 turnoverratio 数值差异达百倍（市盈率多在 10-80 区间，换手率可能低至 0.1% 或高达 20%），必须用 StandardScaler 标准化消除量纲影响——处理后类内紧凑度显著提升，轮廓系数从 0.28 升至 0.47。

确定聚类数量 k 时，通过 “肘部法” 观察 inertia（误差平方和）变化：k=2 到 3 时误差骤降，k=4 时曲线出现明显拐点，故选择 4 类聚类。结果分析颇具业务意义：第一类：低 per（<15）、低 pb（<1.5）、低换手率（<2%），多为金融、公用事业等蓝筹股；第二类：高 per（>50）、高 changepercent（>10%），集中了近期热门的科技成长股；第三类：高换手率（>8%）、pricechange 波动大，符合题材概念股特征；第四类：指标均处于中等水平，以制造业、消费行业的白马股为主。

实操中还解决了两个细节问题：一是通过 dropna 处理缺失值，避免聚类偏差；二是设置 random_state=42 固定初始质心，保证结果可复现。

7 月 26 日 第十二天 星期六

今日学习决策树算法,通过 iris 多分类与波士顿房价预测两个案例，系统掌握了这种树状模型在分类与回归任务中的应用逻辑。决策树以直观的 “if-then” 规则构建预测模型，相比逻辑回归的线性假设，能捕捉特征间的非线性关系，这种特性使其在多类场景中表现尤为灵活。

iris 数据集多分类任务聚焦于如何通过花萼长度、花萼宽度、花瓣长度、花瓣宽度四个特征，区分 Setosa、Versicolor、Virginica 三个鸢尾花品种。实践中，我理解了决策树的分裂逻辑：从根节点开始，每次选择信息增益最大的特征（如花瓣长度≤2.45cm 可完美区分 Setosa)，递归划分样本直至叶节点纯度达标。通过调整 max_depth 参数控制树的复杂度，当深度设为 3 时，模型在测试集上准确率达 97%，既避免了过拟合（深度 5 时准确率下降至 92%)，又保留了足够的分类能力。可视化树结构后发现，花瓣长度是区分鸢尾花的最关键特征。

波士顿房价回归案例则展现了决策树处理连续目标变量的能力。与分类任务不同，回归树以均方误差最小化为分裂准则，通过对特征空间的递归划分，为每个叶节点分配一个平均房价作为预测值。特征重要性分析显示，RM（平均房间数）和 LSTAT（低收入人口比例）对房价影响最大，这与现实中 “地段决定房价” 的规律一致。调优过程中发现，min_samples_split 参数设置为 10 时效果最佳 —— 小于此值会导致过拟合（训练集 R^2 达 0.98，测试集仅 0.71)，大于此值则模型欠拟合（测试集 R^2 降至 0.65)。

通过对比两个案例，我总结出决策树的核心优势：无需特征标准化、可解释性强、能自动处理特征交互。但也存在明显局限：容易过拟合（尤其是样本量较小时)、对噪声数据敏感。解决办法包括剪枝（控制树深度、叶节点样本数)和集成学习(后续将学习的随机森林)。

7 月 27 日 第十三天 星期日

今天学习支持向量机（SVM）算法，通过肥胖风险分类案例，深入理解了这种基于核函数的分类模型的工作原理。SVM 以“寻找最优超平面”为核心思想，与决策树的规则划分不同，它更擅长在高维空间中构建具有最大间隔的分类边界，这种特性使其在小样本、高维度数据中表现突出。

特征预处理环节尤为关键。由于 SVM 对特征尺度敏感，我先对身高（米）、体重（千克）等数值特征做标准化处理，将其转换为均值为 0、标准差为 1 的分布；对性别、家族肥胖史等类别特征采用独热编码，避免引入不合理的数值关系。实践发现，标准化后模型准确率从 0.82 提升至 0.89，这印证了 SVM 对特征量纲的敏感性。

核函数的选择是 SVM 的核心。我对比了三种常用核函数：线性核（linear）在简单线性可分场景表现稳定，但对本案例的复杂特征交互捕捉不足；多项式核（poly）能拟合高阶关系，但参数调试复杂；径向基核（RBF）通过将数据映射到更高维空间，自动捕捉非线性特征，最终在本案例中取得最佳效果（测试集准确率 0.91）。

模型解释方面，通过计算特征权重发现，体重指数（BMI）、每日卡路里摄入量、物理活动频率是影响肥胖风险的三大核心因素，这与医学常识高度一致。可视化支持向量发现，处于分类边界附近的样本多为“临界肥胖”群体，其特征分布更复杂，也正是 SVM 重点关注的决策对象。

今日学习让我认识到，SVM 的优势在于能处理高维特征和复杂非线性关系，但计算复杂度较高，在大数据集上训练效率偏低。明天计划尝试通过特征选择减少维度，进一步优化模型的训练速度与泛化能力。

7 月 28 日 第十四天 星期一

今天开始学习 XGBoost。第五关卡围绕 XGBoost 模型，通过葡萄酒分类与糖尿病指标预测两个案例，深入探究其在多分类和回归任务中的应用奥秘。

葡萄酒分类案例让我明晰了多分类问题的解决路径。该任务是依据葡萄酒的 13 个理化特征（如酒精含量、苹果酸含量、镁含量等），将其分为三个类别。XGBoost 作为集成学习模型，通过构建多棵决策树并组合其结果来进行分类。实践中，我掌握了关键步骤：使用 train_test_split 按 8:2 比例划分训练集和测试集，通过调整学习率、树的深度等参数优化模型。评估时发现，混淆矩阵能清晰展示各类别的预测情况，而准确率、精确率、召回率和 F1 分数等指标的综合考量，能更全面地反映模型在多分类任务中的表现。

糖尿病指标预测案例则呈现了 XGBoost 在回归任务中的应用。与分类不同，此任务是根据患者的年龄、体重指数、血糖浓度等特征，预测其糖尿病的相关指标（如血糖控制情况等连续值），损失函数采用均方误差。特征处理同样重要：对年龄等数值特征进行归一化处理，对性别等类别特征进行标签编码。通过分析不同特征的重要性得分发现，血糖浓度、体重指数和年龄对糖尿病指标的预测影响较大。

今天的学习让我深刻认识到，XGBoost 凭借其高效的并行计算能力和良好的泛化性能，在各类任务中都表现出色。区分多分类与回归任务的关键仍在于目标变量的类型，而参数调优和特征重要性分析则是充分发挥 XGBoost 性能的关键所在。

7 月 29 日 第十五天 星期二

今日迎来机器学习算法综合实战关卡，通过堪培拉天气数据预测任务，将前面积累的模型知识融会贯通。这一关卡不再局限于单一算法，而是要求结合数据特点选择合适模型，并通过完整流程实现预测目标，让我对机器学习实战有了更系统的认知。

堪培拉天气数据集包含日期、温度、湿度、风速、降雨量等多维度特征，核心任务是预测次日是否会下雨（分类问题），同时也可拓展至降雨量等连续值的回归预测。拿到数据后，首要工作是进行全面的探索性分析：通过绘制箱线图发现温度、湿度存在少量异常值，采用 IQR 法进行截断处理；观察特征相关性矩阵，发现湿度与降雨量呈强正相关，温度与风速呈弱负相关。

数据预处理环节比单一案例更复杂：针对日期特征提取季节、月份等时间信息；对缺失值采用分组填充（如按季节填充平均湿度）；对类别特征（如风向）同时尝试独热编码与目标编码，并通过交叉验证对比效果。模型选择上，分别训练了逻辑回归、随机森林、XGBoost 三种模型作为基准，其中 XGBoost 在验证集上的 AUC 值达到 0.87，显著优于其他模型。

进阶优化部分尝试了特征工程创新：将温度、湿度组合生成 "体感指数" 特征，基于风速与风向计算 "风力等级" 衍生变量，使模型性能提升 3%。最后通过 SHAP 值分析发现，前一日降雨量、当日湿度、季节特征是预测下雨的三大关键因素，这与气象学常识高度吻合。

今天的实战让我体会到，真实场景中 "没有最好的算法，只有最合适的流程"—— 从数据理解到模型迭代的每一步都相互影响，而综合运用多种工具的能力，比单独掌握某类算法更重要。这种端到端的实战经验，也让我对机器学习的工程落地有了更具体的认知。

7 月 30 日 第十六天 星期三

今日正式踏入深度学习的领域，第一关的学习聚焦在深度学习的预备知识上，这部分内容繁杂却至关重要，涉及线性代数、概率论、信息论和图模型几个关键方面。

线性代数的内容里，向量运算中的内积和外积，矩阵分解中的特征值分解、SVD 以及高维空间映射等知识点被重点回顾。在深度学习中，权重矩阵的更新其实本质上就是线性变换的优化过程，而激活函数的作用则是通过非线性变换来打破线性不可分的局限。就像图像数据的像素矩阵，通过矩阵运算就能实现卷积操作，这正是卷积神经网络的核心原理所在。

概率论部分围绕着概率分布，像正态分布、伯努利分布，还有期望与方差、最大似然估计等内容展开。在模型训练时，损失函数的设计常常是基于概率假设的，比如交叉熵损失就对应着样本服从多项式分布的假设。另外，dropout 正则化可以理解为对神经元激活概率进行随机调整，通过引入随机性来提升模型的泛化能力。

信息论的核心概念包括熵、交叉熵与 KL 散度。熵是用来衡量信息的不确定性的，比如均匀分布的熵就比正态分布的熵高；交叉熵能够度量两个分布的差异，是分类任务中损失函数的常用形式；KL 散度则可以分解为交叉熵与熵的差值，在生成模型中常用来匹配真实分布与生成分布。

图模型以有向图如贝叶斯网络和无向图如马尔可夫随机场为框架，帮助理解变量间的依赖关系。图模型中的节点代表随机变量，边则表示条件独立性。比如循环神经网络（RNN）的隐藏状态，就可以建模为时序依赖的图结构，每个时刻的状态只与前一时刻相关。
闯关题中的数据预处理与 PCA 任务，要求对高维数据集先进行预处理，包括标准化、缺失值填充，之后再通过 PCA 降维。实践中发现，标准化是 PCA 的前提，要是特征量纲差异大，像年龄与收入，方差大的特征就会主导主成分。通过计算解释方差比，选取前 3 个主成分就能保留 90% 以上的信息，这让我直观感受到降维在减少计算量的同时，还能剔除噪声特征。

而计算图模型的节点度数与熵这一闯关题，针对给定的贝叶斯网络图，需要统计每个节点的入度也就是父节点数和出度也就是子节点数，并计算变量的边际熵。例如“雨天”节点的父节点为“云层”，入度为 1，其熵值会随着天气概率分布的均匀性升高而增大。这个过程让我加深了对图模型结构与信息熵关联性的理解，也让我明白这些预备知识是深度学习后续学习的坚实基础。

7 月 31 日 第十七天 星期四

今日继续深度学习实习部分，关卡 2 聚焦神经网络的原理及其优化方法，内容涵盖深度学习概述、多层感知机、模型的正则化与优化算法，最后通过闯关题在 MNIST 数据集上实践，进一步加深了对这些核心概念的理解。

深度学习概述部分，让我重新梳理了其与传统机器学习的差异——深度学习通过多层非线性变换自动学习特征，无需人工设计特征工程，尤其在图像、语音等复杂数据上表现突出。而多层感知机（MLP）作为最基础的深度学习模型，其结构像多层堆叠的“信息处理器”：输入层接收原始数据（比如 MNIST 的 28×28 像素值），隐藏层通过权重矩阵与激活函数提取抽象特征（从边缘到轮廓再到部件），输出层则映射到分类结果（0-9 的数字）。我注意到，隐藏层的层数和神经元数量需要谨慎设计，过少可能欠拟合，过多则易过拟合且计算成本激增。

模型的正则化是防止过拟合的关键手段。除了之前接触过的 L1、L2 正则化，这里还学到了 Dropout——训练时随机“关闭”部分神经元，迫使模型不依赖特定神经元，增强泛化能力；以及早停法，通过监控验证集损失，在模型开始过拟合前停止训练。这些方法就像给模型“立规矩”，避免它在训练数据上“死记硬背”而失去对新数据的判断力。

优化算法则关乎模型如何高效收敛。从最基础的梯度下降，到带动量的 SGD（模拟物理中的惯性，加速收敛），再到自适应学习率的 Adam（结合动量和 RMSprop 的优势，动态调整每个参数的学习率），不同算法各有侧重。比如 SGD 适合大规模数据，但收敛较慢；Adam 在中小型数据集上往往更快达到较好效果，是实践中常用的选择。

闯关题要求在 MNIST 数据集上训练 MLP，并比较不同激活函数和优化算法的效果。我尝试了 sigmoid、ReLU、tanh 三种激活函数：sigmoid 易出现梯度消失，导致深层网络训练困难；ReLU 能有效缓解梯度消失，训练速度更快，但要注意“死亡 ReLU”问题（部分神经元永久失活）；tanh 性能介于两者之间，但输出均值接近 0，有

时更利于后续层学习。优化算法方面，对比 SGD、Adam 和 RMSprop 后发现，Adam 在相同迭代次数下测试准确率最高（达到 97.5%），且收敛曲线更平稳，而 SGD 需要更多迭代才能接近这一效果。

今天的学习让我意识到，神经网络的性能不仅依赖模型结构，更取决于正则化策略与优化算法的合理选择。激活函数则像“开关”，决定着信息传递的效率，其选择需结合网络深度与任务特性。这些实践经验为后续学习更复杂的网络结构打下了基础。

8 月 1 日 第十八天 星期五

今天学习深度学习基础模型，包括 CNN、RNN 和自动编码器，这些模型在不同的数据处理场景中各有侧重。

CNN（卷积神经网络）最让我印象深刻的是其对图像数据的高效处理能力。它通过卷积层、池化层和全连接层的组合，实现了特征的自动提取。卷积层利用卷积核(过滤器)与输入图像进行滑动卷积，通过局部感受野捕捉图像的局部特征，比如边缘、纹理等，而权值共享则大大减少了参数数量，降低了计算复杂度。池化层则起到降维作用，通过最大值池化或平均池化，在保留重要特征的同时减少数据量。就像处理一张风景照时，卷积核先识别出山脉的轮廓边缘，再通过池化压缩信息，最终由全连接层整合特征进行分类或识别。

RNN（循环神经网络）则专为处理序列数据设计，其核心在于隐藏层的输出会反馈到输入中，形成循环结构，这让它能捕捉时序依赖关系。比如处理文本时，每个单词的理解都依赖于前文的语境，RNN 通过记忆先前的信息来实现这种关联。但普通 RNN 存在梯度消失或爆炸问题，难以处理长序列，而 LSTM（长短期记忆网络）通过引入遗忘门、输入门和输出门，有效解决了这一问题，能更好地保留长期依赖信息，在机器翻译、时间序列预测等任务中表现出色。

自动编码器是一种无监督学习模型，由编码器和解码器两部分组成。编码器将输入数据压缩成低维的潜在表示，解码器则根据潜在表示重构出原始数据。它的核心思想是通过重构误差来学习数据的有效特征，常用于数据降维、去噪和生成新数据。比如对一张带有噪声的图片，编码器提取其核心特征，解码器基于这些特征重建出清晰的图片，过程中自动过滤掉噪声。

闯关题让我动手实践了这些模型的核心操作。对图像进行卷积运算时，我清晰地看到卷积核如何与图像局部区域相乘求和，生成特征图，不同的卷积核会提取出不同的特征，比如垂直边缘和水平边缘。卷积操作的前向传播则让我理解了多层卷积如何逐步组合低级特征形成高级特征。LSTM 单元的前向传播实践中，我跟踪了细胞状态的更新过程，遗忘门决定丢弃哪些旧信息，输入门选择存储哪些新信息，输出门控制输出内容，整个过程像一个精密的 “记忆控制器”。自动编码器的前向传播则让我看到输入数据如何被压缩再重构，当重构结果与输入足够接近时，说明编码器学到了有效的特征表示。

今天的学习让我明白，不同的深度学习模型都是为了适应不同的

数据特性而设计的：CNN 擅长处理网格结构数据（如图像），RNN 适合序列数据（如文本、时间序列），自动编码器则在无监督特征学习中发挥作用。而闯关题的实践让抽象的原理变得具体，每个操作的细节都影响着模型的最终表现，这也让我更加注重对模型底层逻辑的理解。

8 月 2 日 第十九天 星期六

今天是实战，深入学习了 TensorFlow 与 PyTorch 这两大深度学习框架，两者的设计理念差异明显。TensorFlow 采用静态计算图，需要先完整定义网络结构和运算流程，再通过会话启动运行，这种模式在大规模部署时更稳定高效，但调试时不够灵活。而 PyTorch 的动态计算图则方便很多，搭建模型时可以边写代码边运行，随时打印中间变量查看结果，对初学者更友好，尤其适合科研中的快速迭代。

闯关题是用这两个框架完成综合 CNN 与自动编码器的图像分类任务。用 TensorFlow 搭建时，得先定义卷积层、编码解码结构，再配置损失函数和优化器，最后启动会话训练，每一步都要按流程走。换成 PyTorch 时，同样的网络结构写起来更简洁，训练过程中能实时看到特征图的变化，调试时很快就能定位到问题。

实践中发现，处理图像分类任务时，PyTorch 的动态特性让调整网络参数更便捷，而 TensorFlow 在批量处理大量图像数据时速度略快。这次学习让我明白，框架没有绝对好坏，根据任务需求选择合适的工具，才能让效率最大化，也为后续更复杂的项目打下了基础。

8 月 3 日 第二十天 星期日

今天正式步入自然语言处理——NLP 领域,从基础认知到实战应用，对基于 transformers 的 NLP 任务有了系统且深入的理解。

关卡 01 是认识 transformers，这是入门的关键。首先进行了 transformers 的安装,通过 pip 命令轻松完成,随后尝试了基本使用,调用预训练模型进行简单的文本处理，初步感受其强大功能。tokenizers 的使用是重点,了解到不同的 tokenizer(如 BertTokenizer)会将文本分割成子词或字符,这一步直接影响模型的输入质量，实践中发现正确的 tokenize 能让后续模型处理更精准。Bert 模型结构与输入输出解析则让我明白，它采用双向 Transformer 作为编码器，输入包含词嵌入、段嵌入和位置嵌入，输出的隐藏状态蕴含着丰富的上下文信息，为后续任务提供坚实基础。

关卡 02 进行文本分类实战，目标是构建基于 Bert 的企业隐患排查分类模型。先做数据加载与标签分布分析，发现不同隐患类别的样本数量存在差异，这对后续模型训练的平衡有参考意义。数据预处理环节，对企业隐患文本进行清洗、分词，并转化为模型可接受的输入格式。在企业隐患文本分类训练中，利用 Bert 的预训练权重进行微调，通过多次迭代，模型在验证集上的准确率逐步提升，让我体会到预训练模型在特定任务上微调的高效性。

8 月 4 日 第二十一天 星期一

今天继续 NLP。关卡 03 是文本多标签分类实战，基于 Bert 对推特文本进行多标签分类。数据加载后，构建多标签数据集，与单标签不同，这里每个文本可能对应多个标签。模型设置上选取合适的预训练模型，训练过程中采用相应的损失函数。预测与评估时，发现多标签任务的评估指标与单标签有所不同，而阈值选取尤为关键，合适的阈值能平衡精确率和召回率，经过多次试验，找到较优阈值使模型性能更优。

关卡 04 开展句子相似性识别实战，基于 Bert 对句子对进行相似性二分类。首先读取数据并构建自定义数据集对象，将句子对及标签整理成模型所需的结构。模型设置上进行微调，调整参数以适应任务需求。模型预测后，分析错误案例，思考改进思路，比如增加训练数据、调整模型结构等，为提升模型性能提供方向。

关卡 05 进行命名实体识别（NER）实战，基于 Bert 实现文本 NER 任务。先导入基础库并准备数据集，数据集包含文本及对应的实体标签。接着构建 Dataset 与 DataLoader，方便数据的加载和批量处理。模型定义时，在 Bert 基础上添加输出层以适应 NER 任务的标签预测。经过模型训练、验证与评估，查看各项指标，最后进行模型预测，能准确识别出文本中的人名、地名等实体，让我切实感受到 Bert 在 NER 任务中的强大能力。

这两天的学习让我深刻认识到 transformers 尤其是 Bert 模型在 NLP 领域的广泛应用和卓越性能。

8 月 5 日 第二十二天 星期二

今天聚焦于三个更具挑战性的任务：多项选择、文本生成和文本摘要，均基于 Bert 等预训练模型展开，让我对 NLP 的复杂任务处理有了更深入的实践认知。

关卡 06 是多项选择任务实战，目标是基于 Bert 实现 SWAG 常识问题的多项选择。任务介绍中了解到，SWAG 任务主要是给定一个上下文句子，从四个选项中选择最符合常识逻辑的后续句子，这需要模型具备较强的语义理解和常识推理能力。数据集介绍与预处理环节，加载 SWAG 数据集后，发现每个样本包含上下文和四个候选句，预处理时需将上下文与每个候选句组合成输入对，并用 BertTokenizer 进行编码，同时标记正确选项的标签。模型设置上，在 Bert 基础上添加分类层，将 [CLS] token 的输出用于预测每个选项的概率，通过微调让模型学习常识关联，训练过程中观察到验证集准确率稳步提升，逐渐掌握了这类任务的建模思路。

关卡 07 进行文本生成实战，基于预训练模型实现文本生成。任务介绍明确，文本生成旨在根据输入文本生成连贯、相关的输出文本，如续写、问答等。数据集选用了包含对话或故事片段的文本数据，预处理时需对文本进行分段、清洗，并构建输入 - 输出的样本对，确保输入与生成目标的对应性。文本生成模型设置上，选择了适用于生成任务的预训练模型，如 GPT 系列，微调时采用自回归生成的方式，通过调整最大生成长度、温度参数等控制生成效果，实践中发现温度参数对生成文本的多样性影响显著，较低的温度会使输出更确定，较高则更具随机性，反复调试后得到了较流畅的生成结果。

关卡 08 开展文本摘要实战，基于 Bert 实现文本摘要任务。任务介绍指出，文本摘要是将长文本压缩成简洁的摘要，保留核心信息，这要求模型既能理解原文重点，又能进行精炼表达。编码器处理原文获取语义表示，解码器生成摘要，微调过程中重点关注摘要的准确率和简洁性，通过评估指标如 ROUGE 分数来衡量生成效果，经过多轮训练，模型生成的摘要逐渐能抓住原文关键信息，冗余内容减少，让我体会到摘要任务中 “取舍” 的精妙之处。

这一天的学习让我认识到，不同的 NLP 任务对模型的要求各有侧重，多项选择考验推理能力，文本生成注重连贯性与创造性，文本摘要则强调信息提炼，而预训练模型通过微调能较好地适应这些任务，后续还需在调优策略上多下功夫。

8 月 6 日 第二十三天 星期三

今天终于进入 NLP 的尾声了，围绕文本翻译和问答两个实战任务展开，均基于预训练模型进行，让我对这两类任务有了更清晰的认识。

关卡 09 是基于 Bert 的端到端机器翻译。这类任务无需中间步骤，直接实现源语言到目标语言的转换，要求模型掌握双语语义及转换规则。数据集用了双语平行语料，预处理时要分别清洗、分词源语言和目标语言文本，构建文本对并注意长度匹配。模型采用编码器 - 解码器架构，微调时通过交叉熵损失优化，增加 beam search 宽度能提升译文效果，经多轮训练，模型可生成较规范的译文。

关卡 10 为基于预训练模型的 QA 任务，需模型依据上下文回答问题，考验信息定位与提炼能力。数据集用 SQuAD 等，含上下文、问题及答案区间。预处理时组合上下文和问题输入，标记答案位置。模型添加线性层预测答案起止索引，通过精确匹配率等评估，调整上下文长度和问题表述后，准确率有所提升。

这一天的学习让我明白，翻译侧重跨语言语义映射，问答注重信息检索，预处理和调参对模型性能影响很大，后续还需深入探索多语言处理等方面。

8 月 7 日 第二十四天 星期四

今天正式投入到 LoRA 实战项目中，难度远超预期。这次使用的是 OpenDataLab“万卷・丝路 2.0” 语料库，它包含多语言、多模态数据，仅韩语相关的文本及多模态数据就很丰富，本以为能为韩语模型微调提供充足支撑。

学习目标很明确，首要的是理解 LoRA 微调技术的原理和优势。了解到 LoRA 通过冻结预训练模型权重，仅训练低秩矩阵的参数，能大幅减少计算资源消耗，同时保持较好性能。但真正着手处理大规模语言数据时，麻烦接踵而至。

先是在数据处理环节碰壁。“万卷・丝路 2.0” 的韩语数据虽多，却包含多种模态，我需要从中筛选出适合微调的文本数据。尝试用脚本筛选时，多次因数据格式混乱导致程序报错，光是统一数据格式就耗费了半天。好不容易整理好，进行数据清洗时又发现部分文本存在编码问题，有些特殊字符无法正常识别，清洗后的数据集质量远不如预期。

下午开始尝试使用 LoRA 进行模型微调的初期设置。按照教程配置参数，却总是在加载数据集时出现内存溢出的错误。反复调整批次大小，甚至删减了部分数据，才勉强运行起来。但没过多久，训练过程中又出现了梯度爆炸的问题，损失值飙升到无法计算的程度。这一天就在不断报错、排查、修改中度过，别说完成韩语模型微调任务了，连完整的训练流程都没走通，第一次尝试以失败告终。

8 月 8 日 第二十五天 星期五

今天继续尝试攻克LoRA实战项目,带着昨天的挫败感重新出发,没想到失败依旧接踵而至。

吸取了昨天的数据处理教训,今天一早先专注优化数据准备流程。参考了语料库的官方处理指南,发现自己之前忽略了“万卷·丝路2.0”中韩语数据的细粒度分类标签,这些标签本可以帮助筛选更优质的训练数据。重新筛选后,编码问题也通过转换字符集得到解决。原以为数据层面没问题了,可在构建训练集和验证集时,又因划分比例不合理,导致验证集分布与训练集偏差较大,这为后续模型评估埋下隐患。

模型微调环节依旧坎坷。针对昨天的梯度爆炸问题,尝试降低学习率并加入梯度裁剪,这次训练总算能稳定进行一段时间。但新的问题又出现了:训练到第 5 个 epoch 时,模型开始过拟合,训练集损失不断下降,验证集损失却持续上升。尝试增加正则化强度,调整 LoRA 的秩参数,效果都不明显。

更让人沮丧的是,好不容易撑到训练结束,进行模型评估时,发现韩语任务的各项指标都低得离谱。分析日志才发现,由于昨天数据处理时的疏忽,部分文本的标签与内容对应错误,相当于模型一直在“学错东西”。尽管紧急修正后重新训练了一小部分,但时间已经不够,最终的韩语模型微调任务还是以失败收场。不过虽然失败了,但是由于尝试过 LoRA,最后还是通过了。

这两天的经历让我深刻体会到,LoRA 微调看似减少了参数规模,但其对数据质量和流程细节的要求极高,任何一个环节的疏漏都可能导致满盘皆输。失败虽多,但也让我对大规模数据处理和模型微调的复杂性有了更清醒的认识。

8 月 9 日 第二十六天 星期六

今天开始蒸馏任务。由于首次接触知识蒸馏，光是啃理论就耗尽了上午的精力。知识蒸馏的理论基础远比想象中复杂，大语言模型的推理能力涉及逻辑链条的拆解与复现，高级注意力增强技术里的多头注意力机制如何在蒸馏中迁移，这些概念像一团乱麻。尤其是知识路由与专家系统，模型要根据输入动态分配不同专家网络处理，光是理解“门控机制”如何权衡专家权重，就翻了三篇论文还是一知半解。渐进式训练策略里的温度调度、蒸馏损失函数的设计原理，更是让我频频卡壳，总觉得隔着层雾。

下午转入实战，目标是基于教程搭建蒸馏框架。先从镜像配置开始，教程里一句“拉取指定版本镜像”，我却因为没注意 CUDA 版本匹配，连续三次启动容器时出现“驱动不兼容”的报错。好不容易换对镜像，安装依赖包时又陷入版本泥潭——transformers 4.30.0 要求 torch≥2.0.0，可手头的环境里 torch 还是 1.13.1，升级后又发现与 cuda 版本冲突，来来回回卸载重装，直到傍晚才勉强让环境跑通基础测试。

紧接着试跑数据处理模块，面对科学考试任务的数据集，光是解析 JSON 格式就花了一小时。数据里的“问题 - 选项 - 解析”结构需要转换成“教师模型输入 - 学生模型输入 - 蒸馏标签”的格式，可教程里的代码用了自定义的 Dataset 类，其中的__getitem__方法处理逻辑藏得很深。我试着修改字段映射，结果要么出现“key 不存在”的错误，要么标签与输入错位，反复调试到天黑，屏幕上依旧跳着红色报错，第一天连数据加载这关都没过去，更别提触及蒸馏的核心流程了。

8 月 10 日 第二十七天 星期日

早上一来就抱着教程逐行啃数据处理模块的代码，终于发现问题出在编码格式上 —— 数据集里的特殊符号用了 UTF-8-BOM 编码，而代码默认按 UTF-8 读取，导致解析时出现乱码。修正后总算能加载数据，看着终端里滚动的 “加载成功” 提示，心里刚松了口气，新的麻烦又找上门。

搭建知识蒸馏训练器时，师生模型的初始化让我犯了难。教师模型选了 bert-base-uncased，学生模型想用更小的 distilbert-base-uncased，可如何让学生模型的输出层维度与教师模型对齐？教程里用了线性映射层，但我在定义模型时漏算了教师模型的隐藏层维度，导致 forward 过程中出现 “维度不匹配” 的错误。好不容易调好网络结构，启动训练后又发现损失值异常 —— 刚开始就飙升到 1000 以上，而且完全不下降。排查半天才发现，是蒸馏损失函数里的温度参数设反了，本该让教师模型输出软化的温度，被我误用到了学生模型上。

调整参数后重新训练，模型却很快陷入过拟合，训练集准确率飙升到 90%，验证集却始终在 50% 徘徊。尝试换用 BLEU、ROUGE 等指标观察生成质量，发现学生模型只会机械复制教师模型的输出，连简单的句式变换都做不到。想着按作业要求把科学考试任务换成医学考试任务，结果处理数据时彻底卡壳 —— 医学数据里的 “疾病名称 - 症状 - 治疗方案” 结构远比科学题复杂，尤其是带括号的补充说明（如 “高血压（继发性）”），让分词器频频失效，生成的 token 序列混乱不堪。

两天下来，从理论到实战处处碰壁，别说得到能用的学生模型，就连蒸馏过程中温度参数如何影响知识迁移、师生模型的容量配比该怎么设计，依旧一知半解。看着屏幕上最后一次训练的失败日志，真切感受到知识蒸馏这门技术，远比课本里的公式要复杂得多。

8 月 11 日 第二十八天 星期一

今天开始最后的任务: RAG。早上一来就抱着代码逐行啃向量数据库的实现，终于发现问题出在 FAISS 的索引初始化上——IVF_FLAT 索引需要先训练聚类中心，而我直接跳过训练就往里面塞向量，导致报了“训练样本数不足”的错。给训练函数加了动态调整聚类数的逻辑后，总算能正常添加数据，看着终端里打印的“索引构建完成”，心里刚踏实没两分钟，新的麻烦又冒了出来。

对比检索性能时，FAISS 的 IVF_FLAT 结果总不对劲。同样查“退货政策”，原始数据库能精准匹配到“7 天无理由”，IVF_FLAT 却老是返回促销信息。翻了半天文档才明白，小规模数据用 IVF 索引容易聚类偏差，40 多个文本块被硬分成 100 个簇，直接报大错，后来又有些关键词直接被分到了错误的簇里。好不容易把聚类数调到 10，结果又发现检索时的 topk 参数设大了——返回 5 条结果里混了 3 条不相关的，拉低了关键词匹配率。

傍晚试着跑完整对比，结果 IVF_FLAT 的关键词匹配率只有 0.68，比原始数据库低了不少。查阅资料得知 IVF 这种近似索引在数据量小的时候优势体现不出来，反而容易丢精度。想想也是，就几十条数据，全量比对反而更靠谱。

8 月 12 日 第二十九天 星期二

早起开始调 Flat 索引的参数，想看看 FAISS 的精确检索到底比原始数据库强在哪。跑了五组查询后发现，两者的关键词匹配率都是 0.95，但 FAISS 的检索时间从 0.28ms 降到 0.03ms，快了近 10 倍。正高兴呢，突然看到内存占用，这俩其实用的存储空间差不多，因为数据量太小，FAISS 的压缩机制没发挥作用。

接着试分块函数的影响。昨天把 chunk_size 设成 50，结果“分期付款支持 3-24 期”被切成了两半，今天调大到 100，分块倒是完整了，检索时又出了新问题——同一个句子拆成的两块都进了 top3，导致结果重复。改了分块重叠度，把 overlap 从 10 调到 5，重复率才降下去。

下午整理最终结果时发现，IVF_FLAT 虽然快，但在电商场景里不太适用 —— 用户查“商品 A 价格”这种精确需求，差一点都不行。等数据量涨到上万条，也就是大数据量时，IVF 的优势才会出来，现在这规模，Flat 索引足够用了。

两天折腾下来，从索引类型到分块大小，踩的坑比跑通的功能还多。原来向量数据库不是光换个工具就行，还得看数据规模、业务需求，甚至分块时多切一个字都可能影响结果。这实战的水，比课本里的代码深多了。