基于机器学习算法的就业预测和人岗匹配模型

摘要

随着我国科技的进步和经济的发展,就业问题是推动经济发展中的重要研究课题。在考虑人员的个人信息的基础上，本文对影响人员就业情况的指标进行统计和分析，建立了基于机器学习方法的就业分析和预测模型，最终为相关部门制定就业政策提供科学决策依据。

针对问题一，本文首先根据问题说明及附件数据对数据进行特征选取和数据预处理，引入**随机森林模型预测**空缺值，同时结合规则的推算补充和 **MICE 算法**进行填充和异常数据的删改，对缺失值较多且就业关联性不强的数据进行舍弃。编写数据字典，规范化杂乱的毕业学校、户籍地址信息。进而，根据数据间关系，派生出新特征 label、years_since_grad 等。运用 **StandardScaler 标准化缩放**的方法对清洗后的数据进行标准化。将标准化后的数据进行可视化分析、绘制特征相关性热力图、计算**方差膨胀因子 (VIF)** 以检测多重共线性，筛选出与就业相关性较强的特征。

针对问题二，基于问题一中筛选的特征，本文运用问题一的 shap 值特征分析和皮尔逊相关系数热力图初始化各特征对就业影响的初始权重，投入标准化数据，运用 **XGBoost 模型**训练，经过梯度提升迭代机制不断优化权重，准确率、查准率、召回率、F1 等指标对模型进行评估，多轮迭代后通过 gain（特征在每次分裂中带来的平均增益）计算各特征重要性得分，并对重要性排序，得出最优参数后，基于此模型对“预测集”的就业情况进行预测。

针对问题三，本文收集湖北省各地区 2017-2022 六年的高新技术产业增加值与附件人员户籍相匹配，以此来分析经济对就业的影响。我们对这六年的数据求取平均增长率，进行标准化，然后将数据并入我们的训练数据集，再进一步对问题二中的模型进行训练，优化完善 **XGBoost 模型**，基于此模型再次对预测集进行预测。

针对问题四，本文将收集的岗位信息与求职人员的信息作为模型输入。为了能够从文本数据中挖掘出深层次的语义信息，我们引入了先进的自然语言处理技术。具体而言，我们采用了 **SentenceTransformer 模型**，具体是 paraphrase-multilingual -MiniLM-L12-v2 这个预训练模型。该模型具有强大的语义表示能力，能够将文本数据转换为低维向量空间中的稠密向量，使得文本之间的语义相似度可以通过向量之间的距离进行度量，求余弦取相似度。最后根据相似度矩阵，我们为每个求职者生成相应的岗位推荐列表。

最后,我们对提出的模型进行全面的评价：本文的模型贴合实际,能合理解决提出的问题,具有实用性强,算法效率高等特点,该模型在医疗方面也能使用。

关键词：模型预测、特征选择、特征匹配、XGBoost 算法

目录

一、问题重述

1.1 问题背景

就业作为民生之本，是经济社会稳定发展的基石。当前，我国经济正处于高质量发展转型期，就业形势总体稳定，但仍面临结构性矛盾突出、技能供需错配、新兴行业冲击传统岗位等挑战。尤其在“十四五”规划强调“就业优先”战略的背景下，实现高质量充分就业已成为政府宏观调控的核心目标之一。随着人工智能、数字化转型加速，劳动力市场对技能需求快速变化，加之人口老龄化、区域经济发展不均衡等因素叠加，就业问题呈现多维度复杂性。

以宜昌地区为例，其就业生态既受全国性宏观政策影响，也与本地产业结构（如制造业、旅游业等）紧密相关。如何精准分析就业状态的关键驱动因素，预测未来趋势，并为政策制定提供科学依据，成为亟待解决的问题。传统分析方法难以应对多源异构数据（如个人特征、行业动态、宏观经济指标）的融合需求，而大数据与数学建模技术的结合为此提供了新思路。通过构建数据驱动的预测模型，可量化个人特征与宏观因素的交互影响，揭示潜在规律，助力人岗精准匹配与政策优化。

本课题基于宜昌地区 5000 名被调查者的多维度脱敏数据，结合外部经济、政策及市场信息，探索就业状态的分析与预测方法，旨在为地方政府提供动态化、精细化的决策支持，响应新时代“稳就业、促发展”的战略需求。

1.2 问题提出

问题一：根据给定的被调查者当前的就业状态（比如：就业失业时间、录用单位等信息）分析该地区当前就业的整体情况；并将人员按照不同特征进行划分，根据划分特征分析其对就业状态的影响。

问题二：基于问题一的分析，选取与就业状态具有相关性的特征，构建就业状态预测模型并对给定的“预测集”进行预测；并对各特征的重要性进行排序。

问题三：除了个人层面因素影响外，宏观经济、政策、劳动力市场状况、宜昌市居民消费价格指数、招聘信息等也可能会影响就业状态。收集相关数据，提取反映经济、市场等方面的影响因素，并结合问题一中的数据进一步完善就业状态预测模型，并对给定的“预测集”进行预测。

问题四：基于给定的数据，并结合采集到的招聘数据、社交媒体数据、薪资水平、所需技能、宏观经济数据、行业动态数据等建立人岗匹配模型，捕捉求职者和岗位之间的匹配关系，针对给定数据中的失业人员进行工作推荐。

二、问题分析

2.1 问题一的分析

问题一要求分析给定附件人员的就业状态并划分特征对就业状态的影响。首先对数据进行处理，将部分明显没有关联的特征滤去，再对剩余特征进行补充，对于数值型特

征, 使用随机森林模型预测法和 IterativeImputer 实现的 MICE (Multivariate Imputation by Chained Equations) 算法结合进行填充。对数据标准化后，划分特征对于就业状态的影响，进行可视化分析展示。

2.2 问题二的分析

问题二要求基于问题一的分析，选取相关特征构建预测模型并对“预测集”进行预测，同时对各特征重要性进行排序，其关键在于构建预测模型以及确定各特征在模型中的权重占比。在参考文献[1]后本文决定使用 XGBoost 模型对就业情况进行预测，在预设各特征初始权重之后利用数据集不断优化模型，最终得到就业预测模型，同时确定各特征重要性排序。

2.3 问题三的分析

问题三要求收集其他可能影响就业状态的特征进一步完善模型，考虑到高新技术产业是影响经济和就业的一个重要因素，故本文收集近六年宜昌各分区县的高新技术产业增加值与附件数据结合用以优化模型，最终将完善后的模型再次预测“预测集”。

2.4 问题四的分析

问题四要求收集相关岗位的薪资水平、所需技能等信息与求职人员相匹配并进行岗位推荐，参考文献后，本文决定使用 `SentenceTransformer` 模型进行训练，在进行文本预处理之后进行岗位与简历的相似度匹配，并以相似度高低进行岗位推荐。

三、模型假设与符号说明

3.1 模型基本假设

(1) 假设各个特征之间相互独立，即一个特征的取值不会影响其他特征的取值。

(2) 假设数据在时间上具有平稳性，即数据的统计特性不随时间变化而发生显著变化。

(3) 假设模型的误差项是相互独立且服从相同的分布。

3.2 符号说明

表 1 符号说明

符号	含义
$X=[x_1 ,x_2,\cdots,x_n]T$	表示特征矩阵，其中 $x_i=[x_{i1},x_{i2} ,\cdots,x_{im}]$ 是第 i 个样本的特征向量，x_{ij} 表示第 i 个样本的第 j 个特征值。
$y=[y_1 ,y_2 ,\cdots,y_n]T$	表示**目标**向量，其中 y_i 是第 i 个样本的**就业状态标签**，$y_i \in \{0,1\}$，0 表示**失业**，1 表示**就业**。
θ	表示 XGBoost **模型**的**参数集合**，**包括树**的**结构、节点分裂规则、叶子节点**的**权重等**
F1	**精确率和召回率**的**调和平均数**

T	表示 XGBoost 模型进行迭代训练的次数
L(θ)	损失函数
Accuracy	准确率
Precision	精确率
Recall	召回率

注：各参量具体意义将在后文模型建立和求解中进一步明确

四、数据预处理

4.1 特征指标选取：

基于宜昌地区 5000 名被调查者的脱敏数据中的个人基本信息，初步浏览和分析，我们将数据行基本相同的指标（如兵役状态、变动类型、是否独居等）剔除，将数据缺失率较大的指标（如居住信息等）剔除，将部分在常理下与就业情况显然无关的指标去除（如姓名、人员编号），再将反映信息相似的指标择优选取（如所学专业代码和名称、户口所在地代码和名称）。之后根据数据可靠性、完整度、异常率等多维度的指标评判，最终筛选性别（sex）、年龄（age）、民族（nation）、婚姻状态（marriage）、教育程度（edu_level）、政治面貌（politic）、户籍地址（reg_address）、宗教信仰（religion）、文化程度（c_aac011）、毕业学校（c_aac180）、毕业日期（c_aac181）共 11 个指标作为后续模型的训练特征。

4.2 数据清洗：

step1：补全缺失值：

毕业日期缺失：根据人员毕业学校、教育程度等信息，结合各类学校规制，推断就读时长，加上出生日期，得到毕业日期；

分类型数据缺失：编写 fill_multiple_categorical_columns 函数，引入随机森林模型来填补多个缺失的分类特征：

表 2 分类性数据缺失数量及比例

	缺失数量	缺失比例
专业（profession）	1080.0	0.231115
所学专业代码（c_aac182）	739.0	0.158143
所学专业名称（c_aac183）	498.0	0.106570
毕业日期(c_aac181)	31.0	0.006634
户口性质（c_aac009）	11.0	0.002354
文化程度（c_aac011）	1.0	0.000214

step2：去除重复值：

找出人员编号（people_id）中编号相同的数据行，保留其中数据完整度更高或是更新日期更晚的数据行，删除其余相同编号的数据行。

step3：特征转换和派生：

就业状态（label）：新增 label（0,1）标示就业状态（未就业，已就业），引入就业时间、失业时间、失业注销时间、合同终止时间的比较机制进行就业状态推断；

毕业年数（years_since_grad）：通过当前日期和毕业日期计算出毕业年数。

step4：数据规范化：

户籍地址信息（reg_address）：建立各地数据字典，编写 extract_city_or_county 函数，检索地址信息，比对数据字典，对地址数据重新赋值，相同地区（一般以县级行政区划为标准）的地址值相同（如秭归县、长阳土家族自治县等）；

毕业学校（c_aac180）：编写 extract_school_type 函数，检索学校信息，按照学校性质将数据重新赋值为大学、学院、其他。相同性质的学校赋值相同。

step5：数据标准化编码：

使用数据处理的经典方法 sklearn.preprocessing.StandardScaler

$$x_{\text{scaled}} = \frac{x - \mu}{\sigma}$$

将数值特征转换为均值为 0、标准差为 1 的分布，便于后续机器学习模型更好地收敛或优化。

五、模型建立与求解

5.1 问题一模型建立与求解

5.1.1 问题一模型建立

随机森林分类器（Random Forest Classifier）预测缺失的类别变量：

基于多个决策树的投票机制构建的分类器。每棵树对缺失字段进行分类预测，最后投票选出最多的类别。

预测公式：设有 N 棵树 T_1，T_2，...，T_N，对输入特征值 x，每棵树给出一个分类结果：

$$\hat{y} = \text{mode}(T_1(\mathbf{x}), T_2(\mathbf{x}), \ldots, T_N(\mathbf{x}))$$

其中：
mode 表示多数投票的结果。

$x \in R^d$ 是观测的非缺失特征向量

MICE 多重插补（Multiple Imputation by Chained Equations）填补数值型变量中的缺失值：

MICE 是一种链式回归插补方式。每次将一个变量当作目标，其余变量作为输入进行预测，循环更新缺失值，进行迭代预测。

对于每个带缺失变量 X_j，进行如下步骤：

1.使用其他变量 X_{-j} 作为自变量建立线性模型：

$$X_j = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_{j-1} X_{j-1} + \beta_{j+1} X_{j+1} + \cdots + \beta_p X_p + \varepsilon$$

2.基于训练好的模型，对缺失值进行预测，并加入误差项（体现不确定性）：

$$\hat{x}_j^{(i)} = \hat{\beta}_0 + \sum_{k \neq j} \hat{\beta}_k x_k^{(i)} + \varepsilon^{(i)}$$

3.以此更新每个变量缺失值，重复迭代 m 次直到收敛。

Pearson 相关系数（用于热力图）：

皮尔逊相关系数衡量两个连续变量之间的线性相关性，取值范围为 [-1, 1]

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

r = + 1：完全正相关
r = - 1：完全负相关
r =0：无线性相关

方差膨胀因子（VIF）分析：

VIF 用于衡量一个特征与其他所有特征的多重共线性程度。若某变量可以被其他变量高精度预测，说明其共线性强。

$$\mathrm{VIF}_j = \frac{1}{1 - R_j^2}$$

其中 R_j^2 是第 j 个变量对其他变量做回归时的决定系数

VIF < 5 ：共线性可接受

VIF > 10 ：严重共线性，需处理

VIF $\to \infty$：完全线性相关（危险）

逻辑回归（Logistic Regression）：

逻辑回归是一种用于分类问题的广义线性模型（GLM），其核心是预测某事件发生的概率。

$$P(y = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n)}}$$

采用对数似然函数作为目标函数进行最大化：

$$\ell(\beta) = \sum_{i=1}^{m} y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

SHAP（SHapley Additive exPlanations）值量化特征值的边际贡献：

SHAP 基于博弈论中的 Shapley 值思想，量化每个特征对模型预测结果的边际贡献，具有全局和局部解释能力。

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \cdot [f(S \cup \{i\}) - f(S)]$$

即每个特征 i 的贡献等于其加入模型时对预测值提升的平均值，遍历所有可能组合。

5.1.2 问题一模型求解与分析

step1：数据清洗

通过随机森林 Random Forest 预测了户口性质、文化程度、所学专业名称等多种类别变量，因为其能处理非线性、多维度、变量间存在交互的问题；用 MICE 多重插补填补所学专业代码、专业编号等数值型变量中的缺失值，多步迭代，保证预测合理性；再

结合自主编写的函数对毕业年数进行填补，对户籍地址、学校类型进行规范化赋值，数据清洗基本完成。

基于清洗后的数据，统计 label 标签的（0,1）分布，得到就业和失业的人数：

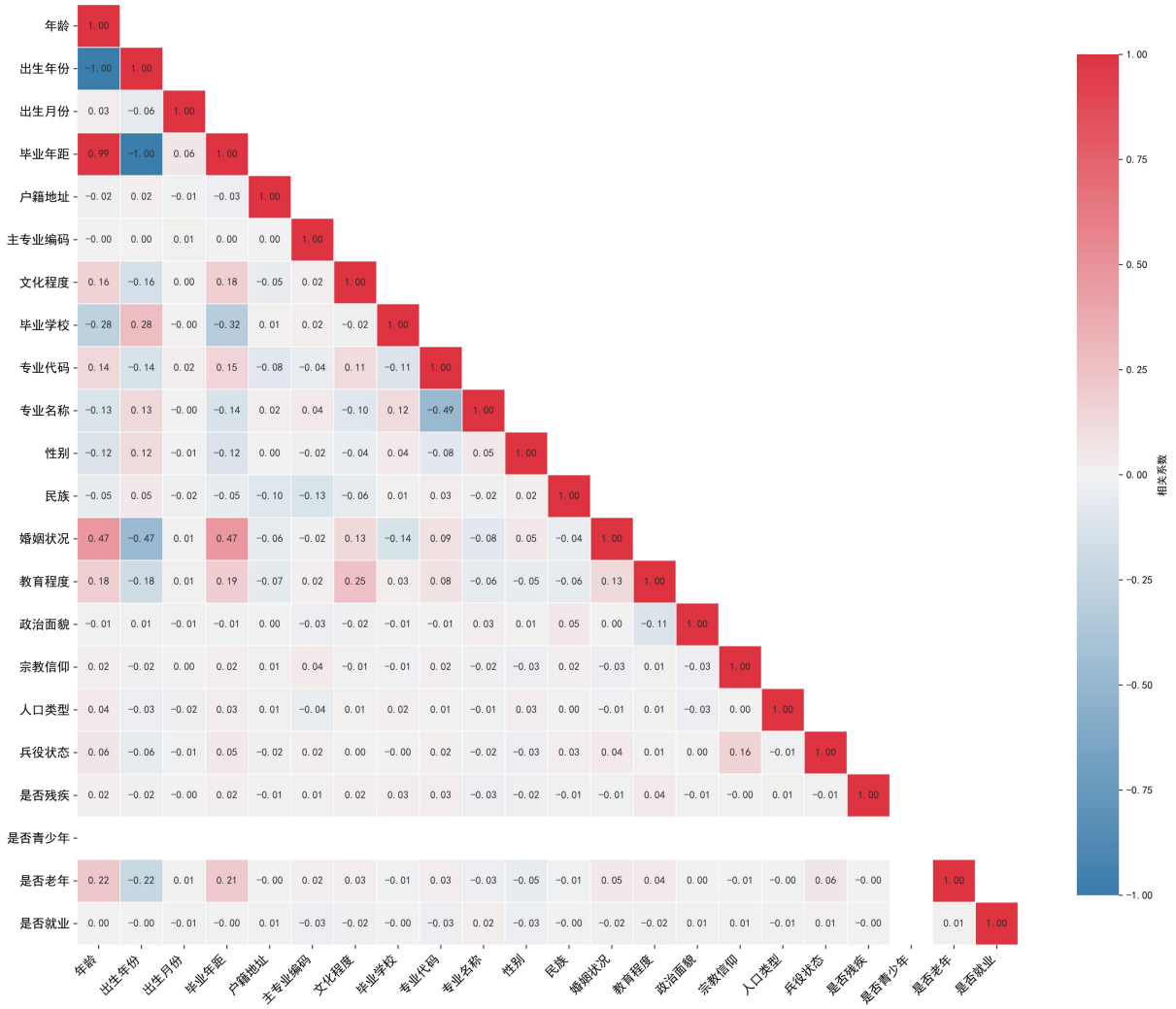
表 3 失业就业人数

就业失业状态	就业	失业
数量（人）	4095	401

step2：相关性分析

利用清洗好的数据，我们对每一对特征变量进行了皮尔逊相关系数的计算，并输出特征矩阵热力图以便对比分析，除了一些常理上相关性较高的变量之外，我们发现婚姻状况、文化程度、专业、性别相较其他特征对就业情况影响比较大，有助于我们进一步分析，同时也可以更好了解数据中各个特征变量之间的相关性，优化特征选取和结构。

特征相关系数矩阵分析



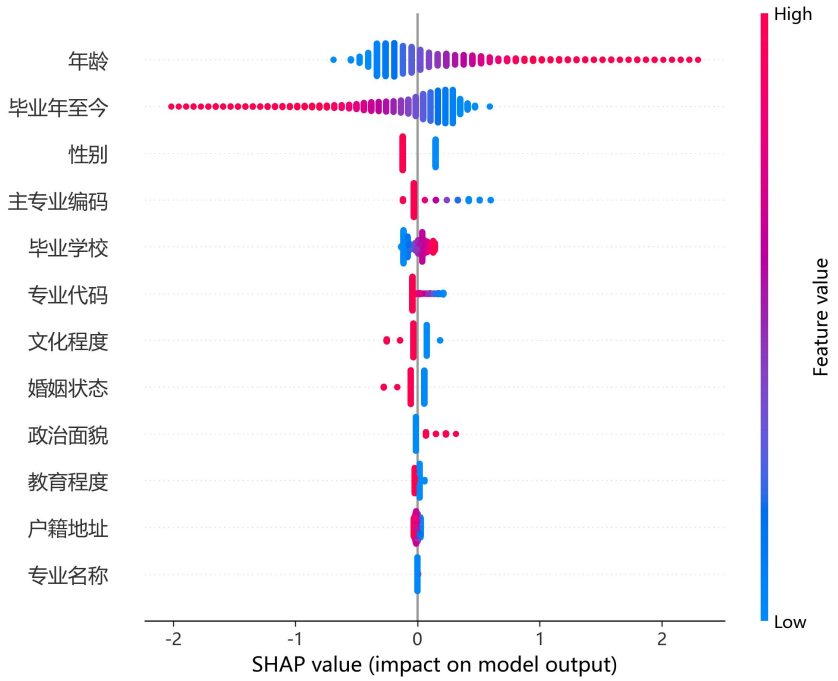
图（1）特征相关系数矩阵热力图

表 4 各特征 VIF 值及共线性评价

特征名称	VIF 值	共线性评价
出生年份	646.171	严重共线性
年龄	501.88	严重共线性
毕业学校	1.54874	严重共线性
出生月份	1.5229	无共线性
专业代码	1.35849	无共线性
专业名称	1.34218	无共线性
婚姻状况	1.33205	无共线性
文化程度	1.15846	无共线性
教育程度	1.12764	无共线性

是否老年	1.062491.	无共线性
民族	1.04148	无共线性
性别	1.04134	无共线性
户籍地址	1.0363	无共线性
兵役状态	1.03543	无共线性
宗教信仰	1.03298	无共线性
主专业编码	1.02724	无共线性
政治面貌	1.01684	无共线性
人口类型	1.00805	无共线性
是否残疾	1.00642	无共线性

再进一步使用 shap 值量化特征值的边际贡献，对选取的特征变量对就业状态模型影响做了可视化输出：



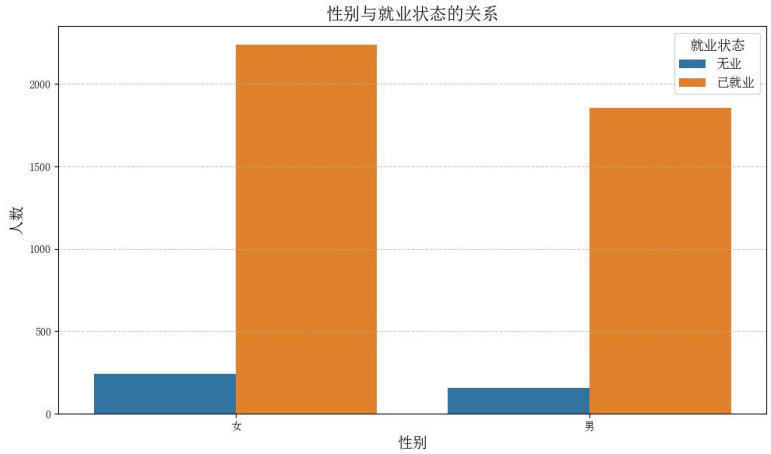
图（2）SHAP value(impact on model output)

如上图所示年龄和毕业年数对就业状况的影响两极分化、且程度最高，说明年龄精力和工作经验对于求职者起着关键的考量作用。剩余特征量对就业状态的影响正负有致，但无明显突出，需要根据 shap 值的关联度进行进一步的模型训练和分析。

根据 shap 值，我们对几大对比度明显的特征值进行统计和可视化分析：

1、性别:

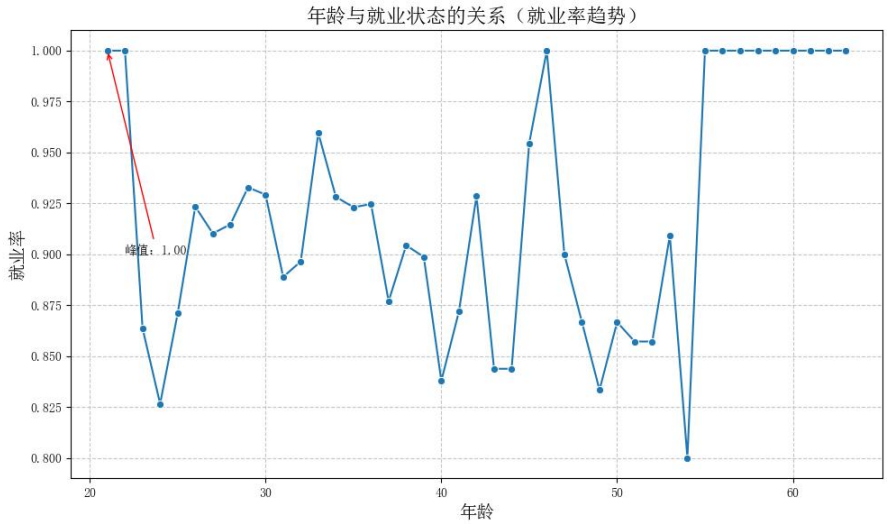
从总量上来看，数据集的女性数量相较男性要多，并且失业和就业人数也相应更多，就就业率而言，女性就业率略高于男性，这可能与当地就业岗位需求有关，这有助于我们进一步优化模型，进行预测。



图（3）性别与就业的关系

2、年龄与就业状态:

就业率随年龄增长宏观呈现“U 型”趋势，25-40 岁青壮年就业率波动不大，稳定在 85%—95%，就业率峰值年龄约为 45-47 岁年龄段（就业率接近 100%），可能与工作经验积累、职业稳定期相关。在这之后显著下降，50 岁后就业率快速下降至约 80%，反映大龄劳动者面临职业转型困难或提前退休现象。给我们以政策启示，需加强中高龄劳动者的技能培训与职业再规划支持，缓解年龄歧视问题。

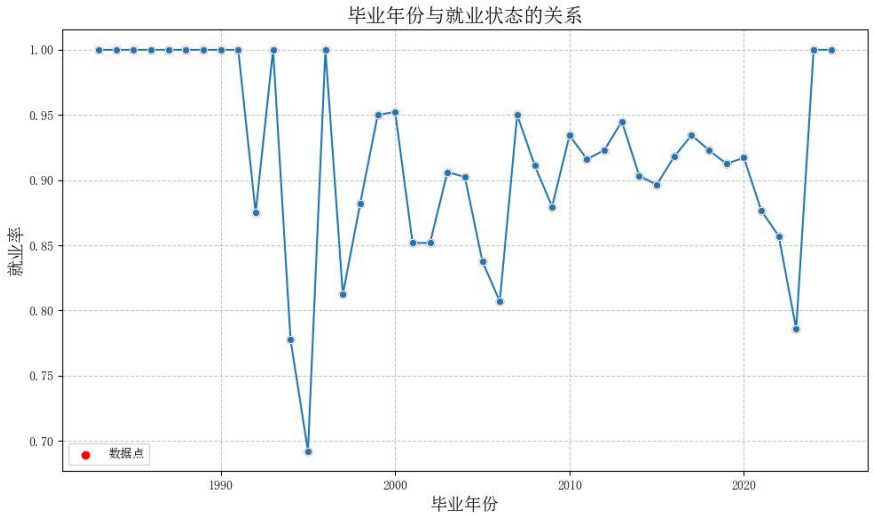


图（4）年龄与就业状态的关系（就业率趋势）

3、毕业年份与就业状态:

毕业年份越近，就业率呈波动下降趋势，且 90 年代波动巨大，可能与经济增速放缓、竞争加剧相关。2005 年后就业率小幅回升（0.90 以上），推测受金融危机后经济复苏政策推动。2020 年又迎来断崖式下滑，2023 年左右触底回弹，重回峰值，可见新冠

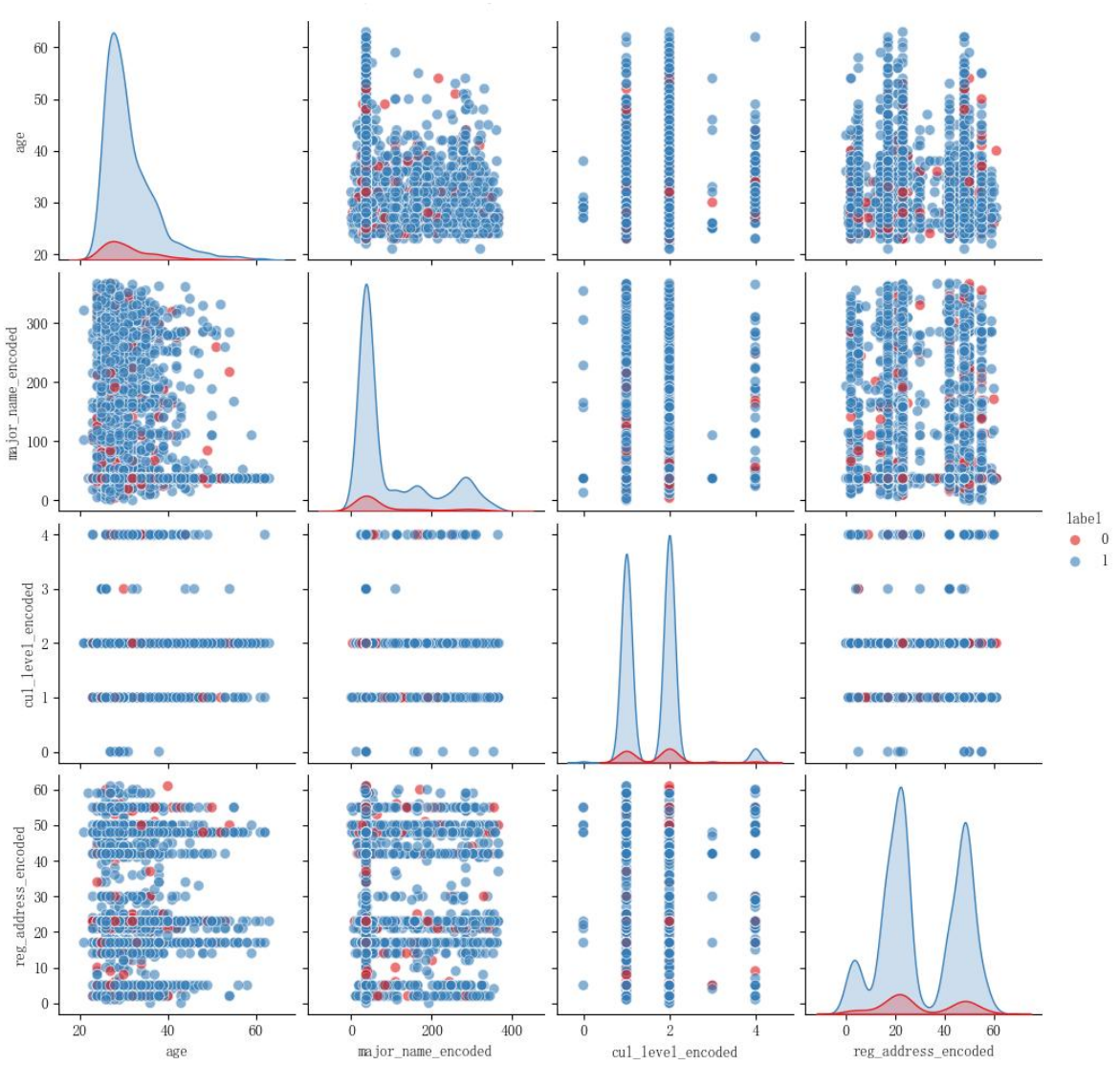
疫情对该地区就业影响极大。纵观时间线，总体就业率还是能维持在 80%以上，可见国家宏观调控经济能力强大。



图（5）毕业年份与就业状态的关系

4、多变量散点化：

进行了年龄（age）、专业、文化程度、户籍地址四对特征变量的分析，通过每对变量之间的散点图，可以初步判断它们之间存在非线性关系，对角线上的核密度估计（kde）图展示每个变量的分布情况，帮助识别变量的分布是否符合常见的模式（如正态分布）。在散点图中，根据离群点的位置，我们可以发现一些异常值或者数据噪声。如下图所示，文化程度和年龄离群点较多，说明大量样本下特例的存在比较普遍，在后续预测时我们可以尽量规避一些特例较多的特征值，更有助于模型预测的精准度。



图（6）多变量散点图

5.2 问题二模型建立与求解

5.2.1 问题二模型建立

1. 数据准备与特征选择

从数据集中选取与就业状态相关的特征。假设我们有 n 个样本，每个样本有 m 个特征，特征矩阵记为 $\mathbf{X}\in\mathbb{R}_{n\times m}$，对应的就业状态标签向量为 $\mathbf{y}=[y_1,y_2,\cdots,y_n]^T$，其中 $y_i\in\{0,1\}$ 表示第 i 个样本的就业状态（0 表示失业，1 表示就业）。

2. 模型目标函数定义：XGBoost 的目标函数由两部分组成：损失函数和正则化项。

损失函数：使用**对数损失函数（Log Loss）**，其表达式为：

$$L(\mathbf{y},\hat{\mathbf{y}}) = \sum_{i=1}^{n}\left[y_i \log(1+e^{-\hat{y}_i}) + (1-y_i)\log(1+e^{\hat{y}_i})\right]$$

其中，$\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_n]^T$是模型的预测值。

正则化项：为了防止过拟合，采用 **L2 正则化**，正则化项的表达式为：

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

其中，f表示一棵决策树，T是决策树的叶子节点数量，w_j是第 j 个叶子节点的权重，γ 和 λ 是正则化参数。

完整目标函数：

在第 t 轮迭代时，**目标函数**为：

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

其中，$\hat{y}_i^{(t-1)}$是前 t−1 轮模型对第 i 个样本的预测值，$f_t(x_i)$是第 t 棵决策树对第 i 个样本的预测值。

3. 模型训练（梯度提升迭代）

采用梯度提升算法进行迭代训练，每一轮迭代构建一棵新的决策树。

泰勒展开近似：

为了求解目标函数，对损失函数进行二阶泰勒展开,忽略常数项第 t 轮的目标函数可以近似为：

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t)$$

叶子节点权重计算：

假设决策树 f_t 将样本划分到不同的叶子节点，设 I_j 是属于第 j 个叶子节点的样本集合，则目标函数可以进一步表示为：

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T$$

对 w_j求偏导并令其为 0，可得第 j 个叶子节点的最优权重为：

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

节点分裂增益计算

为了选择最优的分裂点，需要计算每个特征的每个可能分裂点的增益。分裂增益的计算公式为：

$$\mathcal{G} = \frac{1}{2}\left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda}\right] - \gamma$$

其中，I 是当前节点的样本集合，I_L和 I_R是分裂后左右子节点的样本集合。选择增益最大的分裂点进行分裂。

4. 模型预测

对于样本 **x**，将其输入到训练好的 XGBoost 模型中，通过每棵决策树得到相应的叶子节点权重，然后将所有决策树的叶子节点权重相加，再经过 sigmoid 函数转换为概率值：

$$\hat{y} = \sigma\left(\sum_{t=1}^{T} f_t(\mathbf{x})\right) = \frac{1}{1+e^{-\sum_{t=1}^{T} f_t(\mathbf{x})}}$$

最后根据设定的阈值（0.5）将概率值转换为类别标签：

$$\hat{y}_{label} = \begin{cases} 1, & \hat{y} \geq 0.5 \\ 0, & \hat{y} < 0.5 \end{cases}$$

5. 模型评估

使用准确率、查准率、召回率、F1 等指标对模型进行评估。

准确率（Accuracy）

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

其中，TP 是真正例（实际为就业且预测为就业）的数量，TN 是真反例（实际为失业且预测为失业）的数量，FP 是假正例（实际为失业但预测为就业）的数量，FN 是假反例（实际为就业但预测为失业）的数量。

查准率（Precision）

$$Precision = \frac{TP}{TP+FP}$$

召回率（Recall）

$$Recall = \frac{TP}{TP+FN}$$

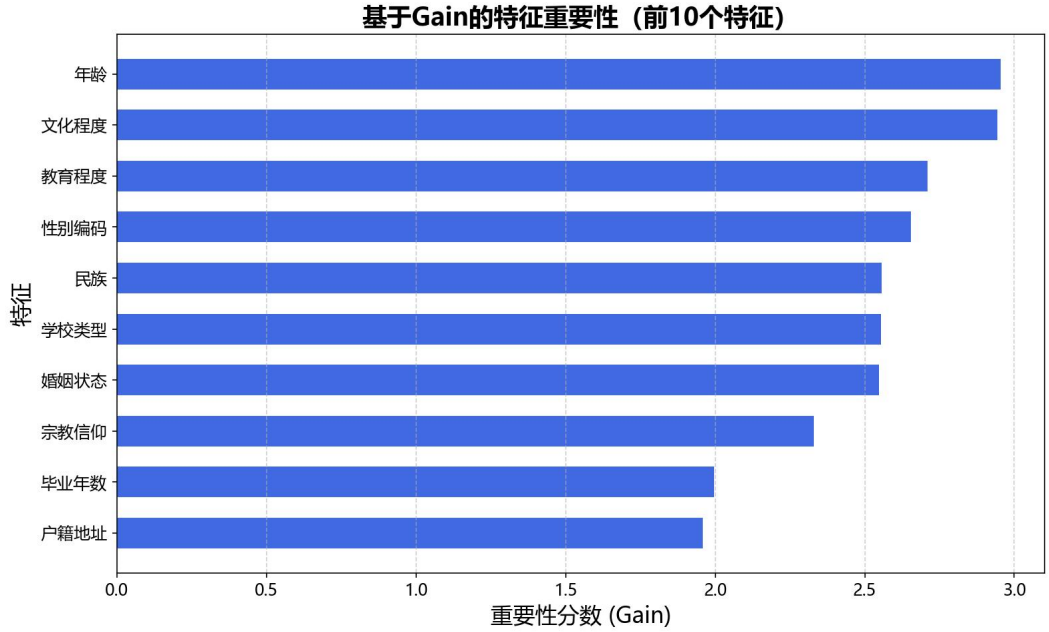
F1 值

$$F1 = \frac{2 \times Precision \times Recall}{Precision+Recall}$$

6. 特征重要性排序

通过 gain（特征在每次分裂中带来的平均增益）计算各特征重要性得分，并对重要性排序。

5.2.2 问题二模型求解与分析



图（7）基于 Gain 的特征重要性（前 10 个特征）

表 5 指标评估

模型	准确率	查准率	召回率	F1
XGBoost	0.92	0.924	1.0	0.96

(保留 2 位小数)

表 6 预测结果

预测	T1	T2	T3	T4	T5	T6	T7	T8	T9
就业状态	1	1	1	1	1	1	1	1	1

预测结果（续）

T10	T11	T12	T13	T14	T15	T16	T17	T18	T19
1	1	1	1	1	1	1	1	1	1

预测结果（续）

T20								就业数量	失业数量
1								20	0

5.3 问题三模型建立与求解

5.3.1 问题三模型建立

模型基本与问题二相同，不再赘述。

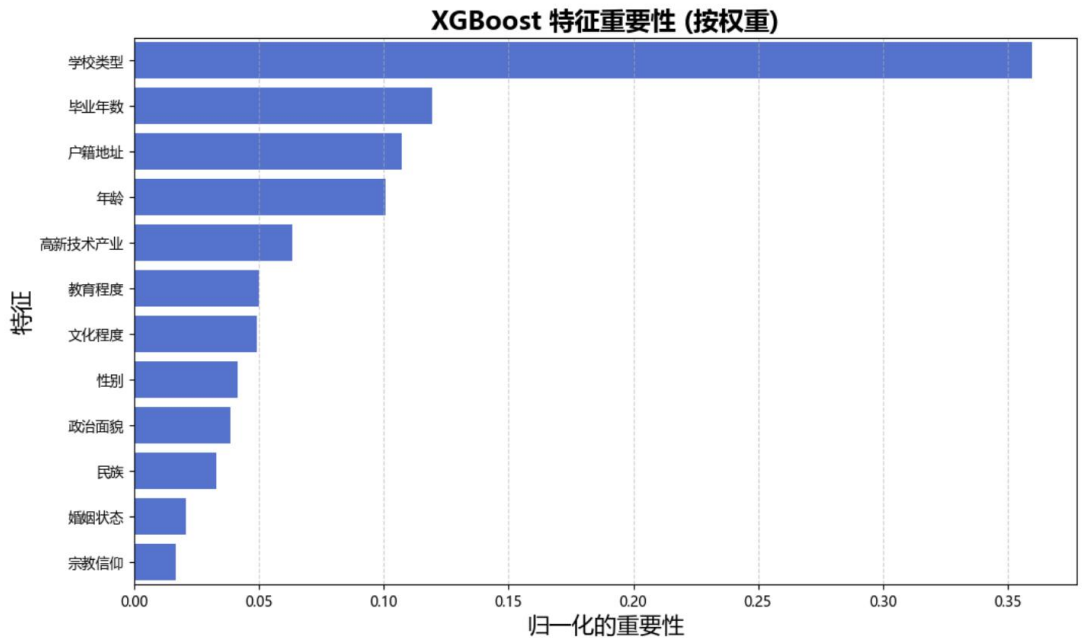
从数据集中选取与就业状态相关的特征。**假设我们有** n 个样本，**每**个样本**有** m 个特征，特征矩阵**记为** $\mathbf{X} \in R_{n\times m}$，**对应**的**就业状态标签**向量**为** $\mathbf{y}=[y_1,y_2,\cdots,y_n]_T$，其中 $y_i \in \{0,1\}$ 表示第 i 个样本的**就业状态**（0 表示**失业**，1 表示**就业**）。

我们通过网络搜集数据，得到 2017 到 2022 年 7 年内湖北省个别地区的高新技术产业增加值数据。我们通过计算每个地区的平均增长率，进行标准化编码方法，然后作为新的数据特征加入到上面的样本特征中，最后使用问题二建立的 XGBoost 模型进行训练以及预测。

$$AverageMean=\frac{1}{5}\sum_{k=1}^{5}(C_{k+1}-C_k)$$

$C_k(k=1,2,3,4,5,6)$ 分别对应 2017 到 2022 这 6 年来的高新技术产业增加值。

5.3.2 问题三模型求解与分析



图（8）CXGBoost 特征重要性（按权重）

经分析：学校类型的特征重要性在众多特征变量里断层式领先，归一化结果占到超过 0.35，说明其在模型训练中的重要性。另外，毕业年数、户籍地址、年龄三个特征变量重要性占比也超过 0.1，也是不可忽略的因素。我们新合并的高新技术产业也印证了我们的预想，成为影响就业状况的关键因素之一，重要性高于一些低阶的基本信息数据，为我们后续预测、人岗匹配以及职位推荐提供更多依据。

表 7 指标评估

模型	准确率	查准率	召回率	F1

XGBoost	0.91	0.91	1.0	0.95

(保留 2 位小数)

表 8 预测结果

预测	T1	T2	T3	T4	T5	T6	T7	T8	T9
就业状态	1	1	1	1	1	1	1	1	1

预测结果（续）

T10	T11	T12	T13	T14	T15	T16	T17	T18	T19
1	1	1	1	1	1	1	1	1	1

预测结果（续）

T20								就业数量	失业数量
1								20	0

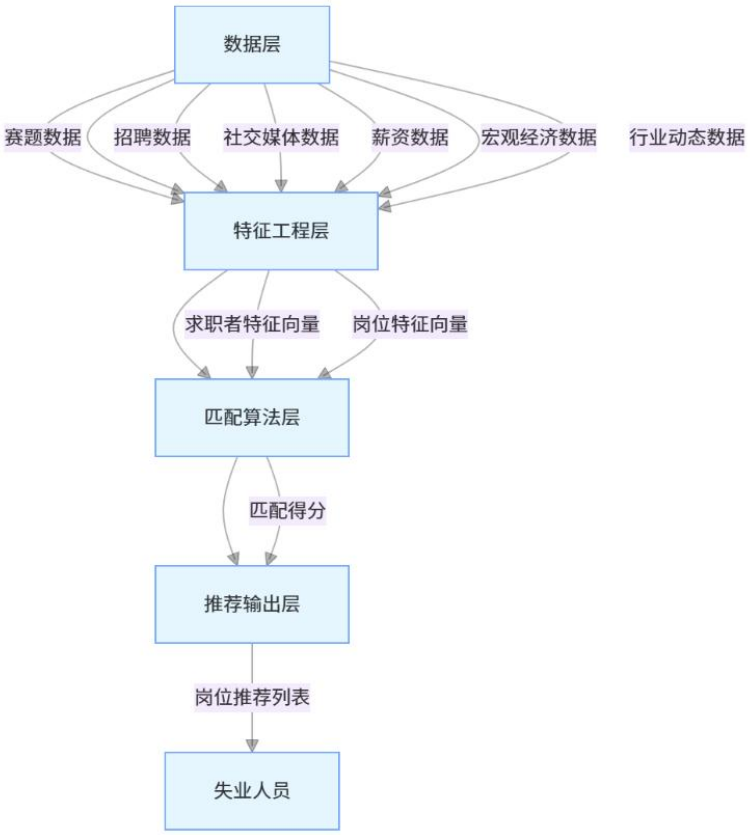
从模型预测结果来看，预测集的就业预测状况比较乐观，经过我们分析，可能与预测集中人员各自具有高学历、学校品牌、身处经济发达地区、毕业年数长、工作经验多等各项优势相关，同时宏观反映湖北各地的经济内生动力充足，高新技术产业发展对就业状况改善效果显著。

5.4 问题四模型建立与求解

5.4.1 问题四模型建立

模型主要包含数据层、特征工程层、匹配算法层和推荐输出层四个部分。数据层整合赛题提供的求职者数据、外部采集的招聘数据、社交媒体数据等多源信息；特征工程层对原始数据进行清洗、转换与特征提取，形成求职者和岗位的结构化特征向量；匹配算法层利用 SentenceTransformer 模型计算文本语义相似度，并结合余弦相似度度量匹配程度；最后，推荐输出层根据匹配得分，为失业人员生成优先级排序的岗位推荐列表。

整体框架如下图：



图（9）模型建立框架图

5.4.2 问题四模型求解与分析

具体模型采用 paraphrase-multilingual-MiniLM-L12-v2 将求职者简历文本和岗位描述文本转换为稠密向量，以捕捉文本的语义信息。该模型基于 Transformer 架构，通过在大规模文本语料上进行无监督预训练，能够学习到语义相似文本的相近向量表示。

再通过相似度计算，将生成的求职者和岗位向量通过余弦相似度计算匹配得分：

$$\text{Similarity}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$$

为减少计算复杂度，采用 Annoy（Approximate Nearest Neighbors Oh Yeah）算法，构建岗位向量索引，快速检索与求职者相似度最高的 Top-N 岗位（本研究设置 N=3）。最终，根据匹配得分对岗位进行排序，为每位失业人员生成个性化推荐列表。

职业推荐结果，限于篇幅只展示 5 个，详细情况见支撑材料:

表 9 职业匹配结果

人员编号	专业	毕业院校	前工作单位	行业编号	岗位推荐
71	学前教育	中华女子学院	东城小学	教育	教师
181	医学检验技术	济宁医学院	夷陵区妇幼保健院	卫生和社会工作	营养师
401	管理类	上海艺术视觉学院	湖北枝江农村商业银行股份有限公司	金融业	会计
998	通信工程	中南民族大学	东城小学	教育	教师
2293	其他学科	三峡大学	五峰土家族自治县公安局	公共管理、社会保障和社会组织	法务专员

六、模型评价与分析

6.1 模型的优点

(1) **模型假设合理，变量选择科学。**利用随机森林 Random Forest 和 MICE 多重插补方法预测填补数据，且进行标准化处理，数据科学严谨。模型在构建过程中充分考虑了多个关键因素，这些变量与就业状况高度相关，能较为全面地刻画影响就业的潜在机制。这样的建模思路贴近实际问题，增强了模型的解释能力和应用价值。

(2) **建模思路清晰，参数设置合理。**模型运用皮尔逊相关系数计算和 shap 值量化法分析特征变量与就业状态的相关性。结合了特征工程与相似度匹配思想，将复杂的就业预测问题简化为特征表达与距离计算的问题。通过对参数的合理设置与调优，模型结构清晰，训练过程稳定，输出结果具有较强的实际指导意义。

(3) **求解算法先进，预测性能优异。**本文采用 XGBoost 算法作为核心预测模型，具备高准确率、鲁棒性强、不易过拟合等优点。配算法层利用 SentenceTransformer 模型计算文本语义相似度，并结合余弦相似度度量匹配程度。同时，算法支持特征重要性评估，便于进一步分析各因素对就业状态的影响，有助于决策解释。

(4) **模型输出稳定，具备实际应用能力。**最终得到的就业安排策略在模拟测试中表现出效率高、波动小、资源分配较为均衡的特点，基本不存在资源浪费、匹配失衡、极端分配等问题，在现有信息条件下能显著提升人岗匹配效率。

6.2 模型的不足

(1) 影响因素考虑不够全面：
在实际应用中，家庭背景、求职意愿、心理状态等主观因素也可能对就业结果产生重要影响，而本文受限于数据条件，未能将其纳入建模范畴，可能一定程度影响模型的泛化能力和预测精度。

(2) 模型适用范围有待拓展：
目前模型在现有就业背景和数据条件下表现良好，但由于时间和资源限制，尚未在其他情形（如区域迁移、行业变动、极端失业情况）中进行充分测试，对模型的稳健性仍需进一步评估。

(3) 变量关系处理仍显简化：
部分变量（如年龄与技能熟练度、学历与薪资期望等）之间的关系在现实中可能是非线性的，但本文在建模过程中将其简化为线性因素处理，忽略了边际效应和变量之间的交互作用，可能影响部分预测精度。

参考文献

[1] 韦园啟. (2022). 基于 ARIMA-XGBoost 模型的中国就业人口预测分析 (硕士 学位论文, 郑州大学). 硕士 https://link.cnki.net/doi/10.27466/d.cnki.gzzdu.2022.000794 doi:10.27466/d.cnki.gzzdu.2022.000794.

[2] 左玉倩. (2022). 基于 BiLSTM 和 XGBoost 的人岗匹配方法研究 (硕士 学位论文, 大连理工大学). 硕士 https://link.cnki.net/doi/10.26991/d.cnki.gdllu.2022.000473 doi:10.26991/d.cnki.gdllu.2022.000473.

[3] 李琦, 孙咏, 焦艳菲, 高岑 & 王美吉. (2019). 基于 HMIGW 特征选择和 XGBoost 的毕业生就业预测方法. 计算机系统应用, 28 (06), 203-208. doi:10.15888/j.cnki.csa.006928.

[4] 杨欣, 刘永喜, 张鑫 & 赵辰. (2023). 基于 GA _ XGBoost 的高校毕业生就业能力预测研究. 软件, 44 (02), 37-41.

附录

工具函数	utils.py
数据预处理	data_preprocess.py
问题 2 数据集数据处理	process_train.py
问题 2 预测集数据处理	process_predict.py
问题 3 数据集数据处理	process_train_final.py
问题 3 预测集数据处理	process_predict_final.py
问题 2 求解	solve2.py
问题 3 求解	solve3.py
问题 4 求解	solve4.py
数据可视化、表格绘制	visualization.py
SHAP 值分析	shap_analysis.py
热力图绘制	p_VIF.py

utils.py

```
import os
import pandas as pd
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
import numpy as np
import re

def load_data(file_path):
    xl = pd.ExcelFile(file_path)

    for i in range(10):
        temp_df = xl.parse(sheet_name='数据集', header=i)
        if 'b_aab022' in temp_df.columns:
            header_row = i
            break
```

```
    else:
        raise ValueError("Error")

    data = xl.parse(sheet_name='数据集', header=header_row)
    print(data.columns.tolist())

    return data

def load_data_pre(file_path):
    xl = pd.ExcelFile(file_path)

    for i in range(10):
        temp_df = xl.parse(sheet_name='预测集', header=i)
        if 'c_aac181' in temp_df.columns:
            header_row = i
            break
    else:
        raise ValueError("Error")

    data = xl.parse(sheet_name='预测集', header=header_row)
    print(data.columns.tolist())

    return data

def load_jobs_code(file_path):
    xl = pd.ExcelFile(file_path)

    for i in range(10):
        temp_df = xl.parse(sheet_name='行业代码', header=i)
        if '行业代码' in temp_df.columns:
            header_row = i
            break
    else:
        raise ValueError("Error")

    data = xl.parse(sheet_name='行业代码', header=header_row)
    print(data.columns.tolist())

    return data

def advanced_missing_value_processing(df):
    missing_analysis = df.isna().agg(['sum', 'mean']).T
    missing_analysis.columns = ['缺失数量', '缺失比例']
    missing_analysis = missing_analysis.sort_values(by='缺失比例', ascending=False)
    print("缺失值分析报告：\n", missing_analysis.head(10))

    # 删除高缺失率特征（阈值设为 70%）
    high_missing_cols = missing_analysis[missing_analysis['缺失比例'] > 0.7].index.tolist()
    if high_missing_cols:
```

```
            print(f"⚠ 删除高缺失率特征：{high_missing_cols}")
            df = df.drop(columns=high_missing_cols)

        # ✓ 毕业年份推算（应放在相关字段填补后）
        if 'c_aac180' in df.columns and 'birthday' in df.columns:
            def infer_grad_year(row):
                if pd.notnull(row.get('graduate_year')):
                    return row['graduate_year']
                school = str(row.get('c_aac180', ''))
                birth_date = row.get('birthday', None)
                if pd.isnull(birth_date):
                    return np.nan
                birth_year = pd.to_datetime(birth_date, errors='coerce').year
                if pd.isnull(birth_year):
                    return np.nan
                # 判断学校类型
                if any(keyword in school for keyword in ['中学', '高中', '职教', '职高', '职工', '
技校', '一中', '二中']):
                    grad_age = 18
                elif any(keyword in school for keyword in ['大学', '学院', '学校']):
                    grad_age = 22
                else:
                    grad_age = 20  # 默认值，适用于无法判断的情况
                return birth_year + grad_age

            df['graduate_year'] = df.apply(infer_grad_year, axis=1)
            df['years_since_grad'] = 2025 - df['graduate_year']

        # 数值型特征处理（MICE 算法）
        numeric_cols = df.select_dtypes(include=np.number).columns.tolist()
        if numeric_cols:
            imputer = IterativeImputer(
                max_iter=10,
                random_state=42,
                initial_strategy='median'
            )
            df[numeric_cols] = imputer.fit_transform(df[numeric_cols])

        return df

address_map_multi = {
        "远安县": ["远安", "河口", "鸣凤", "南门村", "嫘祖", "洋坪", "旧县"],
        "五峰土家族自治县": ["五峰", "渔洋关", "长乐坪", "渔关镇"],
        "恩施土家族苗族自治州": ["恩施", "巴东", "建始", "鹤峰", "利川", "宣恩"],
        "长阳土家族自治县": ["长阳", "白氏坪", "渔峡口", "大堰乡", "龙舟坪"],
        "兴山县": ["兴山", "古夫", "昭君镇", "北斗小区", "梅苑小区"],
        "襄阳市": ["襄阳"],
        "仙桃市": ["仙桃"],
```

```
    "潜江市": ["潜江"],
    "咸宁市": ["咸宁"],
    "天门市": ["天门"],
    "枣阳市": ["枣阳"],
    "孝感市": ["孝感", "孝昌", "汉川"],
    "鄂州市": ["鄂州"],
    "随州市": ["随州"],
    "十堰市": ["十堰", "郧西"],
    "丹江口市": ["丹江口"],
    "神农架林区": ["神农架"],
    "秭归县": ["秭归", "梅家河", "两河口", "茅坪镇", "陈家冲", "平湖", "水田坝", "泄滩
"],
    "枝江市": ["枝江", "白洋", "七星台", "桂溪湖", "龙泉镇", "董市", "马家店", "百里洲",
"安福寺", "仙女镇", "余家溪", "公园路", "民主大道", "建材市场", "九畹溪", "马家街道"],
    "宜都市": ["宜都", "园林大道", "解放社区", "枝城", "陆城", "高坝洲", "赤溪河", "红
湖", "红春", "松木坪", "中笔社区"],
    "西陵区": ["西陵", "港窑", "北苑路", "珍珠路", "发展大道", "城东大道", "东湖大道",
"珠海路", "宜昌市"],
    "伍家岗区": ["伍家岗", "伍临", "东山大道", "夷陵大道", "合益路"],
    "猇亭区": ["猇亭", "下马槽"],
    "夷陵区": ["夷陵", "小溪塔", "罗河路", "分乡", "车站村", "平云", "黄花", "樟村坪", "
乐天溪" ,"下堡坪", "邓村乡", "鸦鹊岭", "雾渡河", "三斗坪", "太平溪"],
    "点军区": ["点军"],
    "荆门市": ["荆门", "钟祥", "沙洋"],
    "武汉市": ["武汉"],
    "当阳市": ["当阳", "坝陵"],
    "荆州市": ["荆州", "公安", "监利"],
    "重庆市": ["重庆"],
    "黄石市": ["黄石", "大冶"],
    "黄冈市": ["麻城", "黄冈", "黄梅", "浠水", "蕲春"],
    "黑龙江省": ["黑龙江"],
    "安徽省": ["安徽"],
    "河南省": ["河南"],
    "甘肃省": ["甘肃"],
    "广东省": ["广东", "深圳", "南山区"],
    "湖南省": ["湖南", "长沙"],
    "内蒙古自治区": ["内蒙古"],
    "四川省": ["四川"],
    "云南省": ["云南"],
    "北京市": ["北京"],
}
def extract_city_or_county(address):
    if pd.isna(address):
        return address
    # 多关键词判断
```

```
    for region, keywords in address_map_multi.items():
        if any(kw in address for kw in keywords):
            return region
    # 正则匹配提取
    match = re.search(r"([^\s]+省)?\s*([^市\s]+市|[^县\s]+县)", address)
    if match:
        return match.group(1)
    match = re.search(r"([^市\s]+市|[^县\s]+县|[^乡\s]+乡|[^镇\s]+镇|[^区\s]+区)", address)
    if match:
        return match.group(1)
    return address
```

data_preprocess.py

```
import os
import pandas as pd
from utils import load_data, load_data_pre, load_jobs_code

data = load_data("../raw_data/附件 1.xls")
data_pre = load_data_pre("../raw_data/附件 1.xls")
job_code = load_jobs_code("../raw_data/附件 1.xls")

data_pre = data_pre.rename(columns={data_pre.columns[-1]: 'label'})

key_columns = ['age', 'sex', 'nation', 'edu_level', 'c_aac009']
data = data.dropna(subset=key_columns, how='any')
data_pre = data_pre.dropna(subset=key_columns, how='any')

data = data.drop_duplicates(subset=['people_id'])
data_pre = data_pre.drop_duplicates(subset=['people_id'])

os.makedirs("../processed_data", exist_ok=True)

data.to_csv(os.path.join("../processed_data/data.csv"), index=False)
data_pre.to_csv(os.path.join("../processed_data/data_pre.csv"), index=False)
job_code.to_csv(os.path.join("../processed_data/job_code.csv"), index=False)

print("Successfully Save to data.csv")
print("Successfully Save to data_pre.csv")
print("Successfully Save to job_code.csv")
```

process_train.py

```
import pandas as pd
from datetime import datetime
import os
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from utils import advanced_missing_value_processing, extract_city_or_county
import joblib
```

```
os.makedirs('../processed_data', exist_ok=True)
os.makedirs('../models', exist_ok=True)

df_raw = pd.read_csv("../processed_data/data.csv", encoding='utf-8')
df = df_raw.iloc[1:].reset_index(drop=True)
df.replace('\\N', pd.NA, inplace=True)

today = pd.to_datetime(datetime.today().date())

date_cols = [
    'b_acc031', #  就业时间
    'b_aae031', #  合同终止日期
    'c_acc028' #  失业注销时间
]
for col in date_cols:
    df[col] = pd.to_datetime(df[col], errors='coerce')

df['label'] = 0
df.loc[df['c_acc0m3'] == '就业', 'label'] = 1
df.loc[df['b_acc031'].notna() & (df['b_aae031'].isna() | (df['b_aae031'] > today)),'label'] = 1
df.loc[df['c_acc028'].notna(), 'label'] = 1

# c_aac009:  户口性质
# c_aab299:  户口所在地区（代码）
# c_aac011:  文化程度
# c_aac180:  毕业学校
# c_aac181:  毕业日期
# c_aac183:  所学专业名称
columns_to_keep = [
    'people_id', 'sex', 'age', 'birthday',
    'nation', 'marriage', 'edu_level', 'politic',
    'reg_address', 'profession', 'religion', 'c_aac009',
    'c_aac011', 'c_aac180', 'c_aac181',
    'type', 'label'
]

df_result = df[columns_to_keep].copy()

# Graduate date
df_result['c_aac181'] = pd.to_datetime(df_result['c_aac181'], errors='coerce')
df_result['graduate_year'] = df_result['c_aac181'].dt.year
df_result['years_since_grad'] = 2025 - df_result['graduate_year']

cat_cols = [
    'sex', 'nation', 'marriage', 'edu_level',
    'politic', 'religion', 'c_aac011', 'reg_address', 'c_aac180']

df_result = advanced_missing_value_processing(df_result)
```

```
df_result['reg_address'] = df_result['reg_address'].apply(extract_city_or_county)
df_result['reg_address'].to_csv('../temp/reg.csv', index=False, encoding='utf-8-sig')

for col in cat_cols:
    le = LabelEncoder()
    df_result[col + '_enc'] = le.fit_transform(df_result[col].astype(str))

# Choosed cols
final_features = ['age', 'years_since_grad'] + [col + '_enc' for col in cat_cols]

# Count duplicate ids
dup_counts = df_result['people_id'].value_counts()
duplicate_ids = dup_counts[dup_counts > 1].index
print(f"Duplicate_ids_count: {len(duplicate_ids)}")

# Remove duplicate ids rows
df_result = df_result[~df_result['people_id'].isin(duplicate_ids)].reset_index(drop=True)
print(f"Removed result: {len(df_result)}")

# Save Non_Standardized_Data
df_result.to_csv('../processed_data/non_standardized_data.csv', index=False,
encoding='utf-8-sig')
print("Successfully save to non_standardized_data.csv")

# Agt2Integer
df_result['age'] = pd.to_numeric(df_result['age'], errors='coerce')

df_model = df_result[final_features + ['label']]

# Standardize
X = df_model[final_features]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
joblib.dump(scaler, '../models/scaler.pkl')
print("Successfully save to scaler.pkl")

X_scaled_df = pd.DataFrame(X_scaled, columns=final_features)
X_scaled_df['label'] = df_model['label'].values

X_scaled_df.to_csv('../processed_data/standardized_data.csv', index=False,
encoding='utf-8-sig')
print("Successfully save to standardized_data.csv")
```

process_predict.py

```
import pandas as pd
from datetime import datetime
import os
import joblib
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
```

```
from utils import extract_city_or_county, advanced_missing_value_processing

os.makedirs('../processed_data', exist_ok=True)

df_raw = pd.read_csv("../processed_data/data_pre.csv", encoding='utf-8')
df = df_raw.iloc[1:].reset_index(drop=True)
df.replace('\\N', pd.NA, inplace=True)

columns_to_keep = [
    'people_id', 'sex', 'age', 'birthday',
    'nation', 'marriage', 'edu_level', 'politic',
    'reg_address', 'profession', 'religion', 'c_aac009',
    'c_aac011', 'c_aac180', 'c_aac181',
    'type']

df_result = df[columns_to_keep].copy()

# Birthday
df_result['birthday'] = pd.to_datetime(df_result['birthday'], errors='coerce')
df_result['birth_year'] = df_result['birthday'].dt.year
df_result['birth_month'] = df_result['birthday'].dt.month


# Graduate date
df_result['c_aac181'] = pd.to_datetime(df_result['c_aac181'], errors='coerce')
df_result['graduate_year'] = df_result['c_aac181'].dt.year
df_result['years_since_grad'] = 2025 - df_result['graduate_year']

cat_cols = [
    'sex', 'nation', 'marriage', 'edu_level',
    'politic', 'religion', 'c_aac011', 'reg_address', 'c_aac180']

df_result = advanced_missing_value_processing(df_result)

df_result['reg_address'] = df_result['reg_address'].apply(extract_city_or_county)

for col in cat_cols:
    le = LabelEncoder()
    df_result[col + '_enc'] = le.fit_transform(df_result[col].astype(str))

# Choosed cols
final_features = [
    'age', 'years_since_grad'] + [col + '_enc' for col in cat_cols]

# Count duplicate ids
dup_counts = df_result['people_id'].value_counts()
duplicate_ids = dup_counts[dup_counts > 1].index
print(f"Duplicate_ids_count: {len(duplicate_ids)}")

# Remove duplicate ids rows
```

```
df_result = df_result[~df_result['people_id'].isin(duplicate_ids)].reset_index(drop=True)
print(f"Removed result: {len(df_result)}")

# Agt2Integer
df_result['age'] = pd.to_numeric(df_result['age'], errors='coerce')

# Standardize
scaler = joblib.load('../models/scaler.pkl')
X = df_result[final_features]
X_scaled = scaler.transform(X)

X_scaled_df = pd.DataFrame(X_scaled, columns=final_features)

X_scaled_df.to_csv('../processed_data/standardized_data_pre.csv', index=False,
encoding='utf-8-sig')
print("Successfully save to standardized_data_pre.csv")
```

process_train_final.py

```
import pandas as pd
from datetime import datetime
import os
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from utils import advanced_missing_value_processing, extract_city_or_county
import joblib

os.makedirs('../processed_data', exist_ok=True)

df_raw = pd.read_csv("../processed_data/data.csv", encoding='utf-8')
df = df_raw.iloc[1:].reset_index(drop=True)
df.replace('\\N', pd.NA, inplace=True)

today = pd.to_datetime(datetime.today().date())

date_cols = [
    'b_acc031', #  就业时间
    'b_aae031', #  合同终止日期
    'c_acc028' #  失业注销时间
]
for col in date_cols:
    df[col] = pd.to_datetime(df[col], errors='coerce')

df['label'] = 0
df.loc[df['c_acc0m3'] == '就业', 'label'] = 1
df.loc[df['b_acc031'].notna() & (df['b_aae031'].isna() | (df['b_aae031'] > today)),'label'] = 1
df.loc[df['c_acc028'].notna(), 'label'] = 1

# c_aac009:  户口性质
# c_aab299:  户口所在地区（代码）
```

```
# c_aac011: 文化程度
# c_aac180: 毕业学校
# c_aac181: 毕业日期
# c_aac183: 所学专业名称
columns_to_keep = [
    'people_id', 'sex', 'age', 'birthday',
    'nation', 'marriage', 'edu_level', 'politic',
    'reg_address', 'profession', 'religion', 'c_aac009',
    'c_aac011', 'c_aac180', 'c_aac181', 'type', 'label']

# 创建高新技术产业增加值数据
data_high_tech = {
    '地区名': ['宜昌市', '宜都市', '枝江市', '当阳市', '远安县', '兴山县', '秭归县', '长阳土家族自治县', '五峰自治县', '夷陵区', '西陵区', '伍家岗区', '点军区', '猇亭区', '宜昌高新区', '恩施土家族苗族自治州'],
    2017: [406.41, 61.12, 43.11, 37.07, 55.94, 4.88, 7.07, 7.77, 2.78, 30.42, 83.92, 9.65, 4.83, 34.42, 23.92, 8.2],
    2018: [472.49, 83.69, 61.24, 55.19, 20.4, 8.11, 10.33, 8.89, 4.12, 44.32, 81.13, 15.57, 9.25, 42.81, 31.27, 5.4],
    2019: [644.65, 119.25, 85.99, 103.57, 14.22, 9.62, 22.01, 17.2, 5.97, 35.5, 92.26, 25.25, 10.36, 53.14, 51.28, 30.7],
    2020: [649.61, 114.78, 106.5, 81.69, 25.41, 8.08, 4.18, 19.23, 6.71, 42.47, 99.44, 25, 6.84, 56.08, 57.69, 29.6],
    2021: [874.15, 152.06, 161, 87.8, 33.63, 12.56, 14.12, 26.08, 10.75, 61.14, 123.15, 35.47, 10.59, 76.64, 75.69, 39.6],
    2022: [1253.7, 243.18, 234.26, 111.05, 49.66, 18.97, 22.88, 26.53, 14.5, 104.83, 148.51, 59.03, 12.43, 126.14, 94.55, 61.1]
}


df_high_tech = pd.DataFrame(data_high_tech)
df_high_tech.set_index('地区名', inplace=True) # 设置地区名字为索引
years = [col for col in df_high_tech.columns if isinstance(col, int)]
growth_rate_df = df_high_tech.copy()
aagr = (df_high_tech[years].iloc[:, 1:].values - df_high_tech[years].iloc[:, :-1].values) /
df_high_tech[years].iloc[:, :-1].values
aagr_mean = aagr.mean(axis=1)
df_high_tech['Average Mean'] = aagr_mean

df_result = df[columns_to_keep].copy()

# Graduate date
df_result['c_aac181'] = pd.to_datetime(df_result['c_aac181'], errors='coerce')
df_result['graduate_year'] = df_result['c_aac181'].dt.year
df_result = advanced_missing_value_processing(df_result)
df_result['years_since_grad'] = 2025 - df_result['graduate_year']
df_result['graduate_year'] = df_result['c_aac181'].dt.year

df_result['reg_address'] = df_result['reg_address'].apply(extract_city_or_county)
```

```
df_result['reg_address'].to_csv('../temp/reg.csv', index=False, encoding='utf-8-sig')

# get mean
def get_value(row):
    region = row['reg_address']
    try:
        return df_high_tech.loc[region, 'Average Mean']
    except KeyError:
        return None

df_result['high_tech'] = df_result.apply(get_value, axis=1)

nan_count = df_result['high_tech'].isna().sum()
print(f"high_tech isnan: {nan_count}")


# remove nan
df_result = df_result.dropna(subset=['high_tech'])

cat_cols = [
    'sex', 'nation', 'marriage', 'edu_level',
    'politic', 'religion', 'c_aac011', 'reg_address', 'high_tech', 'c_aac180']

for col in cat_cols:
    le = LabelEncoder()
    df_result[col + '_enc'] = le.fit_transform(df_result[col].astype(str))

# Choosed cols
final_features = ['age', 'years_since_grad'] + [col + '_enc' for col in cat_cols]

# Count duplicate ids
dup_counts = df_result['people_id'].value_counts()
duplicate_ids = dup_counts[dup_counts > 1].index
print(f"Duplicate_ids_count: {len(duplicate_ids)}")

# Remove duplicate ids rows
df_result = df_result[~df_result['people_id'].isin(duplicate_ids)].reset_index(drop=True)
print(f"Removed result: {len(df_result)}")

# Agt2Integer
df_result['age'] = pd.to_numeric(df_result['age'], errors='coerce')

df_model = df_result[final_features + ['label']]

# Standardize
X = df_model[final_features]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
joblib.dump(scaler, '../models/scaler_final.pkl')
print("Successfully save to scaler_final.pkl")
```

```
X_scaled_df = pd.DataFrame(X_scaled, columns=final_features)
X_scaled_df['label'] = df_model['label'].values

X_scaled_df.to_csv('../processed_data/standardized_data_final.csv', index=False,
encoding='utf-8-sig')
print("Successfully save to standardized_data_final.csv")
```

process_predict_final.py

```
import pandas as pd
from datetime import datetime
import os
import joblib
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from utils import extract_city_or_county, advanced_missing_value_processing

os.makedirs('../processed_data', exist_ok=True)

df_raw = pd.read_csv("../processed_data/data_pre.csv", encoding='utf-8')
df = df_raw.iloc[1:].reset_index(drop=True)
df.replace('\\N', pd.NA, inplace=True)

columns_to_keep = [
    'people_id', 'sex', 'age', 'birthday',
    'nation', 'marriage', 'edu_level', 'politic',
    'reg_address', 'profession', 'religion', 'c_aac009',
    'c_aac011', 'c_aac180', 'c_aac181',
    'type']

df_result = df[columns_to_keep].copy()

data_high_tech = {
    '地区名': ['宜昌市', '宜都市', '枝江市', '当阳市', '远安县', '兴山县', '秭归县', '长阳土家族自治县', '五峰自治县', '夷陵区', '西陵区', '伍家岗区', '点军区', '猇亭区', '宜昌高新区', '恩施土家族苗族自治州'],
    2017: [406.41, 61.12, 43.11, 37.07, 55.94, 4.88, 7.07, 7.77, 2.78, 30.42, 83.92, 9.65, 4.83, 34.42, 23.92, 8.2],
    2018: [472.49, 83.69, 61.24, 55.19, 20.4, 8.11, 10.33, 8.89, 4.12, 44.32, 81.13, 15.57, 9.25, 42.81, 31.27, 5.4],
    2019: [644.65, 119.25, 85.99, 103.57, 14.22, 9.62, 22.01, 17.2, 5.97, 35.5, 92.26, 25.25, 10.36, 53.14, 51.28, 30.7],
    2020: [649.61, 114.78, 106.5, 81.69, 25.41, 8.08, 4.18, 19.23, 6.71, 42.47, 99.44, 25, 6.84, 56.08, 57.69, 29.6],
    2021: [874.15, 152.06, 161, 87.8, 33.63, 12.56, 14.12, 26.08, 10.75, 61.14, 123.15, 35.47, 10.59, 76.64, 75.69, 39.6],
    2022: [1253.7, 243.18, 234.26, 111.05, 49.66, 18.97, 22.88, 26.53, 14.5, 104.83, 148.51, 59.03, 12.43, 126.14, 94.55, 61.1]
}
```

```
df_high_tech = pd.DataFrame(data_high_tech)
df_high_tech.set_index('地区名', inplace=True) #  设置地区名字为索引
years = [col for col in df_high_tech.columns if isinstance(col, int)]
growth_rate_df = df_high_tech.copy()
aagr = (df_high_tech[years].iloc[:, 1:].values - df_high_tech[years].iloc[:, :-1].values) /
df_high_tech[years].iloc[:, :-1].values
aagr_mean = aagr.mean(axis=1)
df_high_tech['Average Mean'] = aagr_mean

df_result = df[columns_to_keep].copy()

# Graduate date
df_result['c_aac181'] = pd.to_datetime(df_result['c_aac181'], errors='coerce')
df_result['graduate_year'] = df_result['c_aac181'].dt.year
df_result = advanced_missing_value_processing(df_result)
df_result['years_since_grad'] = 2025 - df_result['graduate_year']
df_result['graduate_year'] = df_result['c_aac181'].dt.year

df_result['reg_address'] = df_result['reg_address'].apply(extract_city_or_county)
df_result['reg_address'].to_csv('../temp/reg.csv', index=False, encoding='utf-8-sig')

# get mean
def get_value(row):
    region = row['reg_address']
    try:
        return df_high_tech.loc[region, 'Average Mean']
    except KeyError:
        return None

df_result['high_tech'] = df_result.apply(get_value, axis=1)

nan_count = df_result['high_tech'].isna().sum()
print(f"high_tech isnan: {nan_count}")

# Graduate date
df_result['c_aac181'] = pd.to_datetime(df_result['c_aac181'], errors='coerce')
df_result['graduate_year'] = df_result['c_aac181'].dt.year
df_result['years_since_grad'] = 2025 - df_result['graduate_year']

cat_cols = [
    'sex', 'nation', 'marriage', 'edu_level',
    'politic', 'religion', 'c_aac011', 'reg_address', 'high_tech', 'c_aac180']


for col in cat_cols:
    le = LabelEncoder()
    df_result[col + '_enc'] = le.fit_transform(df_result[col].astype(str))

# Choosed cols
final_features = [
```

```
    'age', 'years_since_grad'] + [col + '_enc' for col in cat_cols]

# Count duplicate ids
dup_counts = df_result['people_id'].value_counts()
duplicate_ids = dup_counts[dup_counts > 1].index
print(f"Duplicate_ids_count: {len(duplicate_ids)}")

# Remove duplicate ids rows
df_result = df_result[~df_result['people_id'].isin(duplicate_ids)].reset_index(drop=True)
print(f"Removed result: {len(df_result)}")

# Agt2Integer
df_result['age'] = pd.to_numeric(df_result['age'], errors='coerce')


# Standardize
scaler = joblib.load('../models/scaler_final.pkl')
X = df_result[final_features]
X_scaled = scaler.transform(X)

X_scaled_df = pd.DataFrame(X_scaled, columns=final_features)

X_scaled_df.to_csv('../processed_data/standardized_data_pre_final.csv', index=False,
encoding='utf-8-sig')
print("Successfully save to standardized_data_pre_final.csv")
```

solve2.py

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix
import xgboost as xgb
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import font_manager

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']  # 或者使用 SimHei 等字体
plt.rcParams['axes.unicode_minus'] = False  # 防止负号显示为乱码

data = pd.read_csv('../processed_data/standardized_data.csv')
data_pre = pd.read_csv('../processed_data/standardized_data_pre.csv')
X = data.drop(columns=['label'])
feature_names = X.columns.tolist()
y = data['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
dtrain = xgb.DMatrix(X_train, label=y_train)
dtest = xgb.DMatrix(X_test, label=y_test)
dpre = xgb.DMatrix(data_pre)

params = {
```

```
    'objective': 'binary:logistic',   # 二分类任务
    'eval_metric': 'logloss',         # 评估指标
    'max_depth': 6,                   # 树的最大深度
    'learning_rate': 0.01,            # 学习率
}
num_round = 100

bst = xgb.train(params, dtrain, num_round)

y_pred = bst.predict(dtest)
y_pred_binary = (y_pred > 0.5).astype(int)
mypre = bst.predict(dpre)
mypre_binary = (mypre > 0.5).astype(int)

precision_xgb = precision_score(y_test, y_pred_binary)
recall_xgb = recall_score(y_test, y_pred_binary)
f1_xgb = f1_score(y_test, y_pred_binary)
accuracy_xgb = np.sum(y_pred_binary == y_test) / len(y_test)

print(f"XGBoost 模型-准确率: {accuracy_xgb}")
print(f"XGBoost 模型-查准率: {precision_xgb}")
print(f"XGBoost 模型-召回率: {recall_xgb}")
print(f"XGBoost 模型-F1 分数: {f1_xgb}")

feature_translation = {
    'age': '年龄',
    'years_since_grad': '毕业年数',
    'reg_address_enc': '户籍地址',
    'main_profession_encoded': '主职',
    'c_aac011_enc': '文化程度',
    'c_aac180_enc': '学校类型',
    'major_code_encoded': '专业代码编码',
    'major_name_encoded': '专业名称编码',
    'sex_enc': '性别编码',
    'nation_enc': '民族',
    'marriage_enc': '婚姻状态',
    'edu_level_enc': '教育程度',
    'politic_enc': '政治面貌',
    'religion_enc': '宗教信仰',
    'type_enc': '人口类型',
}

importance_dict = bst.get_score(importance_type='gain')
sorted_importance = sorted(importance_dict.items(), key=lambda item: item[1],
reverse=True)[:10]
```

```
sorted_importance_chinese = [(feature_translation.get(feature, feature), score) for feature, score in sorted_importance]

plt.figure(figsize=(12, 7))
features, scores = zip(*sorted_importance_chinese)
plt.barh(features, scores, color='royalblue', height=0.6)
plt.xlabel('重要性分数 (Gain)', fontsize=16)
plt.ylabel('特征', fontsize=16)
plt.title('基于 Gain 的特征重要性（前 10 个特征）', fontsize=18, fontweight='bold')
plt.gca().invert_yaxis()  # 使得特征按重要性从上到下显示

plt.grid(axis='x', linestyle='--', alpha=0.6)  # 仅对 x 轴添加网格线
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()

cm = confusion_matrix(y_test, y_pred_binary)
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['预测: 0', '预测: 1'], yticklabels=['真实: 0', '真实: 1'])
plt.title("混淆矩阵", fontsize=15, fontweight='bold')
plt.xlabel('预测', fontsize=12)
plt.ylabel('实际', fontsize=12)
plt.show()

cm_df = pd.DataFrame(cm, columns=['预测: 0', '预测: 1'], index=['真实: 0', '真实: 1'])
print("混淆矩阵:\n", cm_df)
```

solve3.py

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score
import xgboost as xgb
import matplotlib.pyplot as plt
import seaborn as sns

plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']  # 或者使用 SimHei 等字体
plt.rcParams['axes.unicode_minus'] = False  # 防止负号显示为乱码

data = pd.read_csv('../processed_data/standardized_data_final.csv')
data_pre = pd.read_csv('../processed_data/standardized_data_pre_final.csv')
X = data.drop(columns=['label'])
feature_names = X.columns.tolist()
y = data['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
dtrain = xgb.DMatrix(X_train, label=y_train)
dtest = xgb.DMatrix(X_test, label=y_test)
dpre = xgb.DMatrix(data_pre)
```

```
params = {
    'objective': 'binary:logistic',   # 二分类任务
    'eval_metric': 'logloss',         # 评估指标
    'max_depth': 6,                   # 树的最大深度
    'learning_rate': 0.01,            # 学习率
}
num_round = 100

bst = xgb.train(params, dtrain, num_round)
score_dict = bst.get_score(importance_type='weight')

y_pred = bst.predict(dtest)
y_pred_binary = (y_pred > 0.5).astype(int)

mypre = bst.predict(dpre)
mypre_binary = (mypre > 0.5).astype(int)

feature_map = {f'f{i}': name for i, name in enumerate(feature_names)}
importance_df_xgb = pd.DataFrame([
    {'feature': feature_map.get(k, k), 'importance': v}
    for k, v in score_dict.items()
])

importance_df_xgb['importance_norm'] = importance_df_xgb['importance'] /
importance_df_xgb['importance'].sum()

precision_xgb = precision_score(y_test, y_pred_binary)
recall_xgb = recall_score(y_test, y_pred_binary)
f1_xgb = f1_score(y_test, y_pred_binary)
accuracy_xgb = np.sum(y_pred_binary == y_test) / len(y_test)

print(importance_df_xgb)
print(f"XGBoost 模型 - 准确率: {accuracy_xgb}")
print(f"XGBoost 模型 - 查准率: {precision_xgb}")
print(f"XGBoost 模型 - 召回率: {recall_xgb}")
print(f"XGBoost 模型 - F1 分数: {f1_xgb}")

plt.figure(figsize=(12, 7))
sns.barplot(x='importance_norm', y='feature', data=importance_df_xgb, color='royalblue')
plt.xlabel('归一化的重要性', fontsize=16)
plt.ylabel('特征', fontsize=16)
plt.title('XGBoost 特征重要性 (按权重)', fontsize=18, fontweight='bold')
plt.grid(True, axis='x', linestyle='--', alpha=0.6)  # 添加网格线
plt.show()
```

solve4.py

```
# -*- coding: utf-8 -*-
import pandas as pd
from datetime import datetime
```

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_curve, auc
import seaborn as sns
import jieba
import re
from collections import Counter
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sentence_transformers import SentenceTransformer


df_raw = pd.read_csv("../processed_data/data.csv", encoding='utf-8')
job_code = pd.read_csv("../processed_data/job_code.csv", encoding='utf-8')

df = df_raw.iloc[1:].reset_index(drop=True)
df.replace('\\N', pd.NA, inplace=True)

today = pd.to_datetime(datetime.today().date())

date_cols = [
    'b_acc031', #  就业时间
    'b_aae031', #  合同终止日期
    'c_acc028' #  失业注销时间
]
for col in date_cols:
    df[col] = pd.to_datetime(df[col], errors='coerce')

df['label'] = 0
df.loc[df['c_acc0m3'] == '就业', 'label'] = 1
df.loc[df['b_acc031'].notna() & (df['b_aae031'].isna() | (df['b_aae031'] > today)),'label'] = 1
df.loc[df['c_acc028'].notna(), 'label'] = 1

job_code = job_code.rename(columns={
    '行业代码': 'b_aab022',
    '注释': 'job_name'})

df = pd.merge(df, job_code[['b_aab022', 'job_name']], how='left', on='b_aab022')
df = df.drop(columns=['b_aab022'])

df_filtered = df[df['label'] == 0]

df_resume = df_filtered[["c_aac180", "c_aac183", "b_aab004", "job_name"]].copy()
df_resume = df_resume.copy()
df_resume.loc[:, '简历文本'] = df_resume.fillna('').astype(str).agg(' '.join, axis=1)
df_resume = df_resume[["c_aac183", "c_aac180", "b_aab004", "job_name", "简历文本"]]
```

```
new_column_names = {
    "c_aac183": "所学专业名称",
    "c_aac180": "毕业学校",
    "b_aab004": "前公司",
    "job_name": "行业名称",
    "简历文本": "简历文本"}

df_resume = df_resume.rename(columns=new_column_names)

# 增加岗位数据
data = [
    {
        "岗位名称": "会计",
        "岗位职责": "全面负责公司财务报表的编制与分析工作，确保报表数据的准确性和及时性，为公司决策提供有力支持。处理日常账务核算，包括审核各类原始凭证、编制记账凭证、登记账簿等。准确计算并及时缴纳各项税费，熟悉税务申报流程和相关税收政策，与税务机关保持良好沟通。定期进行财务数据的统计与分析，为管理层提供财务状况和经营成果的详细报告。协助审计机构完成年度审计工作，提供相关财务资料和解释。",
        "任职要求": "具备会计相关证书，如注册会计师（CPA）、注册税务师（CTA）等优先。熟悉国家财务、税务法规和政策，熟练掌握报税流程和财务软件的使用。具有扎实的会计专业知识和丰富的财务工作经验，能够独立完成财务核算和报表编制工作。具备良好的数据分析能力和财务风险意识，能够发现并解决财务工作中的问题。工作认真负责，严谨细致，具有良好的职业道德和团队合作精神。"
    },
    {
        "岗位名称": "客服专员",
        "岗位职责": "热情、专业地接待用户咨询，通过电话、邮件、在线聊天等多种渠道及时响应用户需求，解决用户问题。详细记录用户反馈的问题和建议，建立用户问题档案，为后续的服务改进提供依据。跟进用户问题的处理进度，及时向用户反馈处理结果，确保用户满意度。积极收集用户对产品或服务的意见和建议，为产品优化和服务提升提供有价值的信息。处理用户投诉和纠纷，以专业的态度和有效的沟通技巧化解矛盾，维护公司良好的品牌形象。",
        "任职要求": "具备良好的沟通能力和语言表达能力，能够清晰、准确地与用户进行交流。拥有高度的耐心和责任心，能够认真倾听用户的诉求，积极解决用户问题。具备较强的服务意识和应变能力，能够在面对用户的不满和投诉时保持冷静，妥善处理。熟悉公司产品或服务的特点和优势，能够为用户提供准确的信息和解决方案。熟练使用办公软件和客服管理系统，具备良好的文字录入和数据处理能力。"
    },
    {
        "岗位名称": "产品经理",
        "岗位职责": "负责公司产品的整体规划和战略制定，结合市场需求和公司发展目标，确定产品的定位和发展方向。组织开展市场调研和竞品分析，深入了解市场动态和用户需求，为产品的优化和创新提供依据。协调公司内部各部门资源，推动产品的开发、上线和迭代更新，确保产品按时交付并达到预期目标。制定产品的营销策略和推广
```

方案，与市场、销售团队紧密合作，提升产品的市场占有率和用户满意度。收集和分析产品的用户反馈和数据指标，持续优化产品的功能和体验，提高产品的竞争力。",
"任职要求": "具备一定的项目管理经验，能够有效地组织和协调团队资源，推动项目的顺利进行。拥有较强的沟通能力和团队协作精神，能够与不同部门的人员进行良好的沟通和合作。具备敏锐的市场洞察力和用户需求分析能力，能够准确把握市场趋势和用户痛点。熟悉产品开发流程和方法，具备良好的产品设计和规划能力。具有较强的数据分析能力，能够通过数据驱动产品决策，优化产品性能。"
},
{
"岗位名称": "UI 设计师",
"岗位职责": "负责公司软件产品的界面设计工作，包括界面布局、色彩搭配、图标设计等，提升用户体验和产品的视觉吸引力。与产品、开发团队紧密合作，深入了解产品需求和用户场景，提供符合用户需求和产品定位的设计方案。进行用户界面的原型设计和交互设计，通过原型演示和用户测试，不断优化设计方案。关注行业动态和设计趋势，将最新的设计理念和方法应用到产品设计中，提升产品的设计水平。参与公司品牌形象设计和宣传物料的设计工作，确保公司品牌形象的一致性和专业性。",
"任职要求": "熟练掌握设计工具，如 Adobe Photoshop、Sketch、Figma 等，具备良好的手绘能力和美术功底。拥有较强的创新能力和审美能力，能够设计出具有独特风格和吸引力的界面。具备良好的沟通能力和团队协作精神，能够与产品、开发团队进行有效的沟通和合作。熟悉用户体验设计原则和方法，能够从用户角度出发进行设计，提高产品的易用性和满意度。具有较强的学习能力和自我驱动力，能够不断学习和掌握新的设计技术和方法。"
},
{
"岗位名称": "销售经理",
"岗位职责": "制定并执行公司的销售策略和销售计划，带领销售团队完成销售目标。负责销售团队的管理和培训，提升团队成员的销售能力和业务水平。建立和维护良好的客户关系，深入了解客户需求，为客户提供优质的产品和服务解决方案。分析市场动态和竞争对手情况，及时调整销售策略和销售计划，提高公司的市场竞争力。组织和参与各类销售活动和商务谈判，拓展销售渠道和客户资源，推动业务的持续增长。",
"任职要求": "具有较强的销售能力和市场开拓能力，能够带领团队完成销售目标。拥有丰富的销售管理经验，能够有效地管理和激励销售团队，提高团队的工作效率和业绩。具备良好的沟通能力和商务谈判技巧，能够与客户建立长期稳定的合作关系。熟悉所在行业的市场情况和客户需求，能够制定针对性的销售策略和解决方案。具有较强的数据分析能力和决策能力，能够通过数据分析为销售决策提供支持。"
},
{
"岗位名称": "HR 专员",
"岗位职责": "负责公司员工的招聘工作，包括制定招聘计划、发布招聘信息、筛选简历、面试安排等，确保招聘到符合公司需求的人才。组织员工培训和发展活动，根据员工的需求和公司的发展战略，制定培训计划并组织实施。负责员工的绩效管理工作，建立和完善绩效管理制度，组织绩效评估和反馈，激励员工提高工作绩效。处理员工关系相关事宜，包括劳动合同管理、劳动纠纷处理、员工关怀等，维护良好的员工关系。参与公司人力资源规划和制度建设，为公司的人力资源管理提供支持和建议。",

"任职要求": "熟悉人力资源管理流程和相关法律法规，具备扎实的人力资源专业知识。拥有良好的沟通能力和组织协调能力，能够与不同部门的人员进行有效的沟通和合作。具备较强的责任心和服务意识，能够认真对待员工的需求和问题，提供优质的人力资源服务。具有较强的数据分析能力和文字处理能力，能够撰写人力资源相关的报告和文件。具备良好的团队合作精神和学习能力，能够不断提升自己的专业水平和综合素质。"

},

{

"岗位名称": "数据科学家",

"岗位职责": "负责公司数据的收集、整理和分析工作，通过数据分析为公司决策提供支持和建议。构建和优化数据模型，运用机器学习、深度学习等算法解决实际业务问题，如预测分析、分类分析、聚类分析等。与业务部门紧密合作，深入了解业务需求，将数据模型和分析结果转化为实际业务解决方案。开发和维护数据平台和数据仓库，确保数据的质量和安全性。跟踪和研究数据科学领域的最新技术和方法，将其应用到公司的业务中，提升公司的数据驱动能力。",

"任职要求": "精通数据分析与机器学习，熟悉常见的机器学习算法和模型，如决策树、神经网络、支持向量机等。具备扎实的数学和统计学基础，能够运用数学方法解决实际问题。拥有丰富的数据分析和建模经验，能够独立完成数据挖掘项目和算法开发。熟练使用编程语言，如 Python、R 等，以及相关的数据处理和分析工具，如 Pandas、NumPy、Scikit-learn 等。具备良好的沟通能力和团队协作精神，能够与业务部门和技术团队进行有效的沟通和合作。"

},

{

"岗位名称": "市场分析师",

"岗位职责": "对市场进行全面深入的调研，包括市场规模、市场趋势、竞争对手情况等，为公司的市场策略制定提供数据支持和分析报告。分析市场数据和信息，运用统计学方法和数据分析工具，挖掘市场机会和潜在风险，为公司的产品定位和营销策略提供建议。跟踪和监测市场动态和竞争对手的活动，及时向公司管理层汇报市场变化情况，为公司的决策提供参考。参与公司的市场推广活动和营销策划，根据市场分析结果制定针对性的推广方案和营销策略。与销售、产品等部门密切合作，共同推动公司业务的发展和市场份额的提升。",

"任职要求": "具备一定的市场分析能力和数据处理能力，能够运用数据分析工具和方法进行市场调研和分析。熟悉市场调研流程和方法，能够设计和执行有效的市场调研方案。拥有良好的数据分析和报告撰写能力，能够将分析结果以清晰、易懂的方式呈现给管理层。具备较强的逻辑思维能力和问题解决能力，能够从复杂的数据中提取有价值的信息和结论。熟悉所在行业的市场情况和竞争态势，能够对市场趋势进行准确的预测和判断。"

},

{

"岗位名称": "客户经理",

"岗位职责": "负责维护和拓展客户关系，与客户建立长期稳定的合作关系，提高客户满意度和忠诚度。深入了解客户需求，为客户提供个性化的产品和服务解决方案，推动销售目标的达成。定期拜访客户，了解客户的使用情况和反馈意见，及时解决客户问题，提升客户体验。制定客户营销计划和方案，通过各种营销手段和活动，促进客户

的二次购买和业务拓展。协调公司内部各部门资源，为客户提供优质的售前、售中、售后服务，确保客户的需求得到及时满足。",

"任职要求": "具备良好的客户沟通技巧和销售经验，能够与客户建立良好的合作关系。拥有较强的服务意识和责任心，能够认真对待客户的需求和问题，提供优质的客户服务。具备较强的市场开拓能力和业务拓展能力，能够不断挖掘新的客户资源和业务机会。熟悉公司的产品和服务，能够为客户提供准确的信息和解决方案。具备良好的团队合作精神和沟通协调能力，能够与公司内部各部门进行有效的沟通和合作。"

},

{

"岗位名称": "网络工程师",

"岗位职责": "负责公司网络设备的配置、维护和管理工作，包括路由器、交换机、防火墙等，确保网络的稳定运行。制定和实施网络安全策略，防范网络攻击和数据泄露，保障公司网络的安全性。进行网络拓扑结构设计和优化，提高网络性能和可靠性。监控网络运行状态，及时发现和解决网络故障，减少网络中断时间。参与公司的信息化建设项目，负责网络部分的规划、实施和验收工作。",

"任职要求": "了解网络协议，如 TCP/IP、HTTP、FTP 等，具备网络设备管理经验。熟悉网络安全技术和方法，能够制定和实施有效的网络安全策略。掌握网络拓扑结构设计和优化的方法，能够提高网络的性能和可靠性。熟练使用网络管理工具和网络诊断工具，能够快速定位和解决网络故障。具备良好的沟通能力和团队协作精神，能够与其他部门的人员进行有效的沟通和合作。"

},

{

"岗位名称": "法务专员",

"岗位职责": "处理公司法务事务，包括合同审查、起草、谈判等，确保合同的合法性和有效性。提供法律咨询服务，为公司各部门提供法律意见和建议，防范法律风险。处理公司的诉讼和仲裁案件，与律师事务所合作，制定诉讼策略，维护公司的合法权益。参与公司的重大决策和项目，从法律角度进行风险评估和合规审查，确保公司的经营活动符合法律法规的要求。开展法律培训和宣传活动，提高公司员工的法律意识和法律素养。",

"任职要求": "法学相关专业，具有扎实的法律专业知识和丰富的法务工作经验。拥有律师资格证者优先，能够独立处理各类法律事务。熟悉国家法律法规和政策，能够准确把握法律风险和合规要求。具备良好的沟通能力和谈判技巧，能够与外部律师事务所和相关部门进行有效的沟通和合作。具有较强的责任心和保密意识，能够认真对待公司的法律事务，保守公司的商业秘密。"

},

{

"岗位名称": "金融分析师",

"岗位职责": "对金融市场进行深入分析，包括宏观经济形势、行业动态、金融产品等，预测未来趋势，为公司的投资决策提供支持和建议。研究和评估金融产品的风险和收益特征，为客户提供专业的投资咨询服务。建立和维护金融模型和数据分析工具，通过数据分析和模型预测，为投资决策提供量化支持。跟踪和监测投资组合的表现，及时调整投资策略，确保投资目标的实现。与其他部门合作，参与公司的金融产品开发和推广工作，提供专业的金融分析和建议。",

"任职要求": "具备金融分析的基本知识，熟悉金融市场和金融产品的特点和风险。熟练使用金融分析工具和软件，如 Excel、MATLAB、EViews 等，能够进行数据分

析和模型构建。拥有丰富的金融分析和投资经验，能够独立完成投资分析报告和投资策略制定。具备较强的数据分析能力和逻辑思维能力，能够从复杂的金融数据中提取有价值的信息和结论。具备良好的沟通能力和团队协作精神，能够与其他部门的人员进行有效的沟通和合作。"
},
{
"岗位名称": "电机结构工程师",
"岗位职责": "进行电机新品结构设计，根据电机的性能要求和应用场景，设计合理的电机结构和外形。负责电机新平台的设计和开发工作，优化电机的结构设计，提高电机的性能和可靠性。与电机研发团队密切合作，参与电机的电磁设计、热设计等工作，确保电机的整体性能符合要求。进行电机结构的强度分析和优化，运用有限元分析软件等工具，对电机结构进行力学性能分析和优化设计。负责电机结构设计文档的编制和管理工作，包括设计图纸、技术文件等，确保设计文档的完整性和准确性。",
"任职要求": "机械设计、电气、材料等相关专业，具有扎实的专业知识和丰富的电机结构设计经验。熟悉电机的工作原理和性能要求，能够根据电机的应用场景进行合理的结构设计。掌握机械设计软件，如 SolidWorks、ProE、UG 等，能够进行三维建模和二维图纸绘制。具备一定的有限元分析能力，能够运用有限元分析软件对电机结构进行力学性能分析和优化设计。具备良好的沟通能力和团队协作精神，能够与其他部门的人员进行有效的沟通和合作。"
},
{
"岗位名称": "PLM 高级产品经理",
"岗位职责": "负责业务调研、需求分析、产品原型设计，深入了解消费电子/大家电/汽车等行业数字化研发体系的业务流程和需求，产出高质量产品说明文档。制定产品的发展战略和规划，结合市场需求和公司发展目标，确定产品的定位和功能特性。协调公司内部各部门资源，推动产品的开发、上线和迭代更新，确保产品按时交付并达到预期目标。与客户和合作伙伴保持密切沟通，收集用户反馈和市场需求，为产品的优化和创新提供依据。组织和参与产品的市场推广和销售活动，提高产品的市场占有率和用户满意度。",
"任职要求": "对消费电子/大家电/汽车等行业数字化研发体系有深入理解，熟悉行业的发展趋势和技术应用。具备丰富的产品管理经验，能够独立完成产品的规划、设计、开发和推广工作。拥有较强的沟通能力和团队协作精神，能够与不同部门的人员进行良好的沟通和合作。具备良好的需求分析和产品设计能力，能够准确把握用户需求和市场趋势，设计出符合用户需求的产品。具有较强的项目管理能力，能够有效地组织和协调团队资源，推动项目的顺利进行。"
},
{
"岗位名称": "高级人力产品经理",
"岗位职责": "参与人力领域薪酬管理、预算管理等产品的规划与搭建，深入了解人力资源管理的业务流程和需求，制定产品的功能和特性。负责产品的需求分析、设计和开发工作，与技术团队合作，确保产品的顺利上线和迭代更新。建立和完善产品的运营体系，包括数据监测、用户反馈收集、产品优化等，提高产品的用户满意度和市场竞争力。与其他部门合作，推动产品的推广和应用，提高公司人力资源管理的数字化水平。跟踪和研究人力资源管理领域的最新技术和趋势，将其应用到产品中，提升产品的创新性和竞争力。",

"任职要求": "具备良好的逻辑思维与沟通表达能力，能够清晰、准确地表达自己的想法和观点。计算机、人力资源管理类专业优先，熟悉人力资源管理流程和相关法律法规。拥有丰富的产品管理经验，能够独立完成产品的规划、设计、开发和运营工作。具备较强的数据分析能力，能够通过数据驱动产品决策，优化产品性能。具备良好的团队合作精神和学习能力，能够不断提升自己的专业水平和综合素质。"

},

{

"岗位名称": "新媒体主管",

"岗位职责": "针对投诉、舆情的用户，查明投诉原因及用户需求，及时、有效地处理用户投诉和舆情事件，维护公司的品牌形象和声誉。制定和执行新媒体营销策略，通过微信、微博、抖音等新媒体平台进行品牌推广和产品宣传，提高公司的知名度和影响力。策划和组织新媒体活动，如线上直播、互动话题等，吸引用户关注和参与，增加用户粘性和活跃度。分析新媒体数据和用户反馈，了解用户需求和市场趋势，优化新媒体营销策略和内容。管理新媒体团队，包括人员招聘、培训、绩效考核等，提高团队的工作效率和业务水平。",

"任职要求": "逻辑思维清晰，具备较强服务意识和应变能力，能够在面对突发舆情事件时保持冷静，妥善处理。熟悉新媒体平台的运营规则和特点，能够制定针对性的新媒体营销策略。拥有丰富的新媒体运营经验，能够独立完成新媒体活动的策划、组织和执行工作。具备良好的沟通能力和团队协作精神，能够与不同部门的人员进行有效的沟通和合作。具备较强的数据分析能力，能够通过数据分析优化新媒体运营效果。"

},

{

"岗位名称": "一线服务专员",

"岗位职责": "负责销售、金融保险、产品等业务的推广和销售工作，向客户介绍产品的特点和优势，促成交易。为客户提供基础售后服务咨询，解答客户关于产品使用、保养、维修等方面的问题，提高客户满意度。收集客户反馈和市场信息，及时向公司反馈客户需求和市场动态，为公司的产品优化和服务提升提供依据。处理客户投诉和纠纷，以专业的态度和有效的沟通技巧化解矛盾，维护公司良好的品牌形象。协助团队完成销售目标和任务，积极参与团队的各项活动和培训。",

"任职要求": "具备良好的服务意识和沟通能力，能够热情、专业地与客户进行交流。熟悉销售、金融保险、产品等业务的基本知识和特点，能够为客户提供准确的信息和解决方案。拥有较强的销售能力和市场开拓能力，能够完成销售目标和任务。具备良好的应变能力和问题解决能力，能够在面对客户的不满和投诉时保持冷静，妥善处理。具备良好的团队合作精神和学习能力，能够不断提升自己的业务水平和综合素质。"

},

{

"岗位名称": "室内设计师",

"岗位职责": "负责室内空间的方案设计，包括功能规划、空间布局、风格定位等，满足客户的个性化需求。绘制施工图纸，包括平面图、立面图、效果图等，确保设计方案的准确传达与实施。与施工团队密切配合，进行施工现场的技术指导，解决施工过程中的设计问题。协助客户进行材料选型和家具配饰的搭配，把控整体设计效果。跟进项目进度，确保项目按时、高质量完成，维护良好的客户关系。",

"任职要求": "室内设计、环境艺术设计等相关专业优先，具备扎实的设计理论基础。熟练掌握 AutoCAD、3DMAX、Photoshop 等设计软件，能够独立完成设计方案的绘制与表现。具有丰富的室内设计项目经验，熟悉施工工艺和材料特性。具备良好的

沟通能力和审美能力，能够准确理解客户需求，将创意转化为实际设计方案。具有团队协作精神和责任心，能够适应一定的工作压力和项目周期。"
    },
    {
        "岗位名称": "软件开发工程师",
        "岗位职责": "参与软件项目的需求分析和设计，根据业务需求制定技术方案。负责软件代码的编写、调试和维护，确保代码质量和系统稳定性。进行软件系统的测试和优化，修复发现的问题，提高系统性能和用户体验。与团队成员协作，共同完成项目开发任务，及时沟通解决开发过程中的技术难题。关注行业技术动态，引入新技术和方法，提升产品的竞争力。",
        "任职要求": "计算机科学与技术、软件工程等相关专业，本科及以上学历优先。熟练掌握至少一种编程语言，如 Java、Python、C++等，熟悉常用开发框架和工具。具有良好的编程习惯和代码规范意识，具备较强的问题解决能力和逻辑思维能力。有软件开发项目经验，熟悉软件开发流程，能够独立完成模块开发。具备良好的团队合作精神和沟通能力，能够适应快速变化的开发环境。"
    },
    {
        "岗位名称": "营养师",
        "岗位职责": "为客户进行营养状况评估，根据个人身体状况、饮食习惯和健康目标，制定个性化的营养方案。开展营养知识讲座和健康咨询活动，向客户普及科学的饮食营养知识。与医疗机构、健身中心等合作，为特殊人群（如孕妇、老年人、慢性病患者等）提供专业的营养指导和饮食建议。跟踪客户的营养方案实施效果，定期进行回访和调整，确保达到预期健康目标。参与营养产品的研发和推广，提供专业的技术支持。",
        "任职要求": "营养学、食品科学、公共卫生等相关专业，持有营养师资格证书。熟悉各类食物的营养成分和营养价值，掌握常见疾病的营养干预方法。具备良好的沟通能力和服务意识，能够与客户建立良好的信任关系。具有较强的学习能力和创新意识，关注营养领域的最新研究成果和发展趋势。有一定的数据分析能力，能够对客户的营养状况进行科学评估和跟踪。"
    },
    {
        "岗位名称": "物流专员",
        "岗位职责": "负责物流运输计划的制定和执行，协调货物的收发、仓储和配送工作。跟踪货物运输状态，及时处理运输过程中的异常情况，确保货物按时、安全送达。与供应商、运输公司等保持良好沟通，优化物流渠道，降低物流成本。管理物流单据和信息，保证物流数据的准确记录和及时更新。参与仓库管理，协助进行库存盘点和货物整理，提高仓储效率。",
        "任职要求": "物流管理、交通运输等相关专业优先，了解物流行业的运作流程和规范。熟悉物流信息系统操作，具备良好的数据分析和处理能力。具有较强的沟通协调能力和问题解决能力，能够有效应对物流过程中的突发情况。工作认真负责，具备良好的时间管理能力和团队合作精神。有一定的抗压能力，能够适应物流行业的快节奏工作。"
    },
    {
        "岗位名称": "教师",

"岗位职责": "根据教学大纲和课程标准，制定教学计划，精心设计教学内容和教学方法。承担课堂教学任务，运用多样化的教学手段，激发学生的学习兴趣和积极性。批改学生作业，进行学习辅导和答疑，及时了解学生的学习情况并给予反馈。组织和指导学生参加各类学科竞赛、课外活动，培养学生的综合素质和创新能力。与家长保持沟通，反馈学生在校表现，共同促进学生成长。参与教学研究和课程开发，不断提升教学质量和专业水平。",
"任职要求": "师范类相关专业或所授学科相关专业，持有教师资格证书。具备扎实的专业知识和教育教学理论基础，熟悉教育心理学知识。具有良好的语言表达能力和课堂组织能力，能够生动有效地进行教学。热爱教育事业，富有责任心和爱心，关心学生的全面发展。具备较强的学习能力和团队协作精神，能够适应教育改革和发展的需求。"
},
{
"岗位名称": "摄影师",
"岗位职责": "根据拍摄任务和客户需求，制定拍摄方案，包括拍摄主题、风格、场景和构图等。负责各类题材的拍摄工作，如人像、产品、风景等，运用摄影技术和艺术手法，捕捉高质量的影像。进行照片的后期处理和编辑，包括色彩调整、修图、合成等，提升作品的艺术效果。管理摄影设备和器材，定期进行维护和保养，确保设备正常运行。与客户、模特、化妆师等密切合作，营造良好的拍摄氛围，保证拍摄任务顺利完成。",
"任职要求": "摄影、视觉传达等相关专业优先，具备扎实的摄影理论知识和实践经验。熟练掌握各类摄影设备和后期处理软件，如 Adobe Lightroom、Photoshop 等。具有独特的艺术视角和审美能力，能够创作出具有创意和感染力的摄影作品。具备良好的沟通能力和团队协作精神，能够与不同人员有效合作。有较强的责任心和敬业精神，能够适应不同的拍摄环境和工作强度。"
},
{
"岗位名称": "厨师",
"岗位职责": "负责厨房菜品的研发和创新，根据市场需求和客户口味，推出新菜品。制定每日菜单，合理安排食材采购和库存管理，控制成本。组织和指导厨房团队进行食材加工、烹饪制作，确保菜品质量和出餐速度。监督厨房卫生和安全操作规范，保持厨房环境整洁，预防食品安全事故。参与厨房设备的维护和管理，及时提出设备更新和维修需求。",
"任职要求": "烹饪专业或相关职业培训经历，持有厨师职业资格证书。精通各类烹饪技法，熟悉不同菜系的特点和制作工艺。具有丰富的厨房管理经验，能够合理安排工作流程和人员分工。具备良好的食材鉴别能力和成本控制意识，保证菜品品质和经济效益。有较强的团队管理能力和沟通能力，能够带领团队完成各项工作任务。"
}
]

```
df_jobs = pd.DataFrame(data)
df_jobs['岗位文本'] = df_jobs[['岗位名称', '岗位职责', '任职要求']].agg(' '.join, axis=1)

model = SentenceTransformer("../models/paraphrase-multilingual-MiniLM-L12-v2")
```

```
resume_embeddings = model.encode(df_resume["简历文本"].tolist(),
show_progress_bar=True)
job_embeddings = model.encode(df_jobs["岗位文本"].tolist(), show_progress_bar=True)

similarity_matrix = cosine_similarity(resume_embeddings, job_embeddings)

top_n = 3
recommendations = []

for i, sims in enumerate(similarity_matrix):
    top_indices = sims.argsort()[-top_n:][::-1]
    top_jobs = df_jobs.iloc[top_indices]["岗位名称"].tolist()
    recommendations.append(top_jobs)

df_resume["岗位推荐 Top{}".format(top_n)] = recommendations
df_resume.to_csv("../results/job_result.csv", index=False, encoding='utf-8-sig')
```

visualization.py

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib
from matplotlib import font_manager
import os

font_path = "C:\\Windows\\Fonts\\simsun.ttc"  # 或者使用 SimSun

if os.path.exists(font_path):
    my_font = font_manager.FontProperties(fname=font_path)
    matplotlib.rcParams['font.family'] = my_font.get_name()  # 设置全局字体
    matplotlib.rcParams['axes.unicode_minus'] = False  # 防止负号显示为方框
    print(f"已设置中文字体为: {my_font.get_name()}")
else:
    print("字体未找到，请检查路径！")

df = pd.read_csv('../processed_data/non_standardized_data.csv')

sex_mapping = {0: '未知', 1: '男', 2: '女', 9: '未说明'}
df['sex'] = df['sex'].map(sex_mapping).fillna('未知')

edu_mapping = {
    10: '研究生教育', 11: '博士研究生毕业', 12: '博士研究生结业', 13: '博士研究生肄业',
    14: '硕士研究生毕业', 15: '硕士研究生结业', 16: '硕士研究生肄业', 17: '研究生班毕
业',
    18: '研究生班结业', 19: '研究生班肄业', 20: '大学本科教育', 21: '大学本科毕业',
    22: '大学本科结业', 23: '大学本科肄业', 28: '大学普通班毕业', 30: '大学专科教育',
    31: '大学专科毕业', 32: '大学专科结业', 33: '大学专科肄业', 40: '中等职业教育',
    41: '中等专科毕业', 42: '中等专科结业', 43: '中等专科肄业', 44: '职业高中毕业',
```

```
    45: '职业高中结业', 46: '职业高中肄业', 47: '技工学校毕业', 48: '技工学校结业',
    49: '技工学校肄业', 50: '高中以下', 60: '普通高级中学教育', 61: '普通高中毕业',
    62: '普通高中结业', 63: '普通高中肄业', 70: '初级中学教育', 71: '初中毕业',
    73: '初中肄业', 80: '小学教育', 81: '小学毕业', 83: '小学肄业', 90: '文盲或半文盲',
    91: '中等师范学校（幼儿师范学校）毕业', 92: '中等师范学校（幼儿师范学校）结业
',
    93: '中等师范学校（幼儿师范学校）肄业', 99: '其他'
}
df['edu_level'] = df['edu_level'].map(edu_mapping).fillna('未知')

min_grad_year = df['graduate_year'].min()
df_filtered = df[df['graduate_year'] > min_grad_year]

output_dir = '../figure'
os.makedirs(output_dir, exist_ok=True)

employment_rate_by_grad_year = df_filtered.groupby('graduate_year')['label'].mean()
employment_rate_by_age = df.groupby('age')['label'].mean()


# 1. 性别与就业状态
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='sex', hue='label', hue_order=[0, 1])
plt.title('性别与就业状态的关系', fontsize=16)
plt.xlabel('性别', fontsize=14)
plt.ylabel('人数', fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.legend(title='就业状态', labels=['无业', '已就业'], fontsize=12, title_fontsize=13,
loc='upper right')
plt.tight_layout()
plt.savefig(os.path.join(output_dir, 'sex_vs_employment_status.png'))
plt.close()

# 2. 教育程度与就业状态
plt.figure(figsize=(12, 10))
sns.countplot(data=df, y='edu_level', hue='label', hue_order=[0, 1])
plt.title('教育程度与就业状态的关系', fontsize=20)
plt.xlabel('人数', fontsize=14)
plt.ylabel('教育程度', fontsize=14)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.legend(title='就业状态', labels=['无业', '已就业'], fontsize=12, title_fontsize=13)
plt.tight_layout()
plt.savefig(os.path.join(output_dir, 'edu_level_vs_employment_status.png'))
plt.close()

# 3. 年龄与就业率折线图
plt.figure(figsize=(10, 6))
```

```
sns.lineplot(x=employment_rate_by_age.index, y=employment_rate_by_age.values,
marker='o')
max_idx = employment_rate_by_age.idxmax()
plt.annotate(f'峰值：{employment_rate_by_age.max():.2f}', xy=(max_idx,
employment_rate_by_age.max()),
                 xytext=(max_idx + 1, employment_rate_by_age.max() - 0.1),
                 arrowprops=dict(arrowstyle='->', color='red'))
plt.title('年龄与就业状态的关系（就业率趋势）', fontsize=16)
plt.xlabel('年龄', fontsize=14)
plt.ylabel('就业率', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.savefig(os.path.join(output_dir, 'age_vs_employment_rate.png'))
plt.close()

# 4. 毕业年份与就业率
plt.figure(figsize=(10, 6))
sns.lineplot(data=employment_rate_by_grad_year, marker='o')
plt.scatter(employment_rate_by_grad_year.index, employment_rate_by_grad_year.values,
            color='red', label='数据点')
plt.title('毕业年份与就业状态的关系', fontsize=16)
plt.xlabel('毕业年份', fontsize=14)
plt.ylabel('就业率', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend()
plt.tight_layout()
plt.savefig(os.path.join(output_dir,
'graduate_year_vs_employment_status_with_markers.png'))
plt.close()

# 5. 年龄与就业状态箱线图
plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x='label', y='age', hue='label', palette='coolwarm')
plt.title('年龄与就业状态的箱线图', fontsize=16)
plt.xlabel('就业状态', fontsize=14)
plt.ylabel('年龄', fontsize=14)
plt.xticks([0, 1], ['无业', '已就业'], fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.savefig(os.path.join(output_dir, 'age_vs_employment_boxplot.png'))
plt.close()

# 6. 性别与就业状态饼图
sex_label_counts = df.groupby(['sex', 'label']).size().unstack().fillna(0)
fig, axes = plt.subplots(1, 2, figsize=(12, 6))
labels_dict = {0: '无业', 1: '已就业'}

for i, col in enumerate(sex_label_counts.columns):
```

```
    axes[i].pie(sex_label_counts[col], labels=sex_label_counts.index,
                autopct='%1.1f%%', startangle=140, textprops={'fontsize': 10})
    axes[i].set_title(f'就业状态：{labels_dict.get(int(col), str(col))}', fontsize=14)

plt.suptitle('性别与就业状态的分布（饼图）', fontsize=16)
plt.tight_layout()
plt.savefig(os.path.join(output_dir, 'sex_vs_employment_pie.png'))
plt.close()

print("所有图表已保存至 'figure' 文件夹")
```

shap_analysis.py

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import shap
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

import matplotlib
matplotlib.rcParams['font.sans-serif'] = ['Microsoft YaHei']
matplotlib.rcParams['axes.unicode_minus'] = False

plots_dir = "../figure"
os.makedirs(plots_dir, exist_ok=True)

data_path = "../processed_data/standardized_data.csv"
try:
    data = pd.read_csv(data_path)
except FileNotFoundError:
    raise FileNotFoundError(f"✖ 未找到数据文件：{data_path}")

data['是否在职'] = data['label']

features = [
    'age', 'years_since_grad', 'sex_enc', 'nation_enc',
    'marriage_enc', 'edu_level_enc',
    'politic_enc', 'religion_enc',
    'reg_address_enc']

X = data[features]
y = data['是否在职']

# 划分训练测试集并拟合逻辑回归模型
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
model = LogisticRegression(solver='liblinear')
model.fit(X_train, y_train)
```

```
# 生成 SHAP 值
explainer = shap.Explainer(model, X_train, feature_names=features)
shap_values = explainer(X_train)

rename_dict = {
    'age': '年龄',
    'years_since_grad': '毕业年至今',
    'sex_enc': '性别',
    'nation_enc': '民族',
    'marriage_enc': '婚姻状态',
    'edu_level_enc': '教育程度',
    'politic_enc': '政治面貌',
    'religion_enc': '信仰',
    'reg_address_enc': '户籍地址'
}
readable_features = [rename_dict.get(f, f) for f in features]

shap.summary_plot(shap_values, features=X_train, feature_names=readable_features,
show=False)
plt.tight_layout()
plt.savefig(os.path.join(plots_dir, "shap_summary_plot_filtered.png"), dpi=300)
plt.close()

print("✓ SHAP 特征重要性图已保存：figure/shap_summary_plot_filtered.png")
```

p_VIF.py

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

feature_rename_map = {
    'age': '年龄',
    'sex_enc': '性别',
    'nation_enc': '民族',
    'birth_year':'出生年份',
    'marriage_enc': '婚姻状况',
    'edu_level_enc': '教育程度',
    'politic_enc': '政治面貌',
    'religion_enc': '宗教信仰',
    'type_enc':'人口类型',
```

```
    'years_since_grad': '毕业年距',
    'label': '是否就业'
}

def load_and_validate_data(path):
    try:
        df = pd.read_csv(path, encoding='utf-8-sig')
        print(f"成功加载数据，维度：{df.shape}")

        assert 'label' in df.columns, "数据中缺少必需的 label 列"
        assert df.shape[0] > 100, "数据量过少（行数<100），请检查数据文件"

        return df
    except Exception as e:
        print(f" 数据加载失败：{str(e)}")
        raise


def generate_correlation_heatmap(df):
    # 计算相关系数
    corr_matrix = df.corr(numeric_only=True)

    corr_matrix = corr_matrix.rename(index=feature_rename_map,
columns=feature_rename_map)

    # 创建 mask 矩阵
    mask = np.triu(np.ones_like(corr_matrix, dtype=bool), k=1)

    plt.figure(figsize=(20, 18))
    cmap = sns.diverging_palette(240, 10, s=80, l=50, as_cmap=True)

    # 绘制相关系数矩阵（热力图）
    ax = sns.heatmap(
        corr_matrix.round(2),
        mask=mask,
        cmap=cmap,
        annot=True,
        fmt=".2f",
        center=0,
        square=True,
        linewidths=0.8,
        cbar_kws={"shrink": 0.8, "label": "相关系数"},
        annot_kws={"size": 10, "color": "#2d2d2d"}
    )

    ax.set_title("特征相关系数矩阵分析\n",
                 fontsize=24,
                 fontweight='bold',
                 pad=25)
```

```
    plt.xticks(rotation=45, ha='right', fontsize=12)
    plt.yticks(rotation=0, fontsize=12)

    # 保存输出
    output_path = "../figure/correlation_matrix.jpg"
    plt.savefig(output_path, dpi=600, bbox_inches='tight')
    plt.close()
    print(f"\n 相关系数矩阵图已保存至：{output_path}")

def perform_vif_analysis(df, vif_threshold=10):
    X = df.drop(columns=['label'])
    X_const = add_constant(X)

    vif_results = []
    for i in range(X_const.shape[1]):
        feature_name = X_const.columns[i]
        try:
            vif = variance_inflation_factor(X_const.values, i)
        except Exception as e:
            print(f" 计算特征 '{feature_name}' 时发生错误：{str(e)}")
            vif = np.inf
        vif_results.append(vif)

    vif_df = pd.DataFrame({
        '特征名称': X_const.columns,
        'VIF 值': vif_results,
        '共线性评价': pd.cut(
            vif_results,
            bins=[0, 5, 10, 50, np.inf],
            labels=["无共线性", "轻度共线性", "显著共线性", "严重共线性"],
            right=False
        )
    }).sort_values('VIF 值', ascending=False)

    # 筛选建议
    high_vif_features = vif_df[vif_df['VIF 值'] > vif_threshold]['特征名称'].tolist()
    if high_vif_features:
        print("\n 高共线性特征建议：")
        print(" | ".join(high_vif_features))

    return vif_df

def generate_markdown_report(report_data):
    try:
        md_content = f"""
# 数据特征分析报告
**生成时间**：{datetime.now().strftime("%Y-%m-%d %H:%M:%S")}
```

```
## 1. 目标变量分布

{report_data['label_dist']}

## 2. 特征共线性分析（VIF）

{report_data['vif_table'].to_markdown(index=False)}

**共线性说明**：
- 严重共线性（VIF>50）：红色标记特征
- 显著共线性（VIF>10）：橙色标记特征
- 轻度共线性（VIF>5）：黄色标记特征

## 3. 特征相关性热力图
![相关系数矩阵](./figure/correlation_matrix.jpg)

## 4. 分析建议

### 数据质量问题
{report_data['quality_issues']}

### 特征工程建议
1. **共线性处理**：
    - 删除特征：`{', '.join(report_data['high_vif_features'])}`
    - 使用 PCA 降维（保留 85%方差）
2. **特征优化**：
    - 离散化处理：`birth_month` 转换为季度特征
    - 嵌入编码：`main_profession_encoded` 使用 FastText 编码

### 建模建议
- 样本重采样（SMOTE 过采样）
- 使用 LightGBM 内置类别特征处理
- 添加特征交互项：`school_encoded × major_code_encoded`
            """

            report_path = "../processed_data/reports/analysis_report.md"
            with open(report_path, 'w', encoding='utf-8') as f:
                f.write(md_content)

            print(f"\nMarkdown 分析报告已生成：{report_path}")
            return report_path

        except Exception as e:
            print(f"\n 报告生成失败：{str(e)}")
            raise

if __name__ == "__main__":
```

```
try:
    df = load_and_validate_data('../processed_data/standardized_data.csv')

    report_data = {}

    label_dist = df['label'].value_counts().reset_index()
    label_dist.columns = ['就业状态', '数量']
    report_data['label_dist'] = label_dist.to_markdown(index=False)

    vif_report = perform_vif_analysis(df)

    vif_report['特征名称'] = vif_report['特征名称'].replace(feature_rename_map)

    vif_report['特征名称'] = vif_report.apply(
        lambda x: f"<span style='color:red'>{x['特征名称']}</span>"
        if x['VIF 值'] > 50 else (
            f"<span style='color:orange'>{x['特征名称']}</span>"
            if x['VIF 值'] > 10 else x['特征名称']
        ), axis=1
    )

    report_data['vif_table'] = vif_report

    high_vif = vif_report[vif_report['VIF 值'] > 10]
    report_data['high_vif_features'] = high_vif['特征名称'].tolist()
    report_data['quality_issues'] = "- 检测到 {} 个高共线性特征（VIF>10）".format(
        len(report_data['high_vif_features'])
    )

    generate_correlation_heatmap(df)

    generate_markdown_report(report_data)

    # 保存最终数据集
    selected_features = [
        'sex_enc',
        'nation_enc',
        'marriage_enc',
        'edu_level_enc',
        'politic_enc',
        'religion_enc',
        'label'
    ]

    output_path = "../processed_data/final_processed_data.csv"
    df[selected_features].to_csv(output_path, index=False, encoding='utf-8-sig')
    print(f"\n 最终数据集已保存至：{output_path}")

except Exception as e:
```

```
print(f"\n 执行过程中发生严重错误：{str(e)}")
print("建议检查：")
print("1. 输入文件路径是否正确")
print("2. 数据是否包含非数值型列")
print("3. Python 依赖库版本是否兼容")
```