基于信用卡客户交易数据的精准智能营销研究

摘　要

信用卡作为便捷支付工具，随着经济和科技的发展，市场迅速扩展。然而，近年来信用卡市场增速放缓。为了留住存量客户和利用大数据进行精准营销，激发消费潜力，我们通过建立**基于 PCA 与随机森林的客户分期意愿预测模型**、**熵权法—TOPSIS 模型**、**基于组合赋权法的 RFMN 模型**等模型去解决分期意愿评分、客户综合价值评估等问题。

针对问题一，对所给数据进行预处理，删除重复用户 id 数 414 例，然后对数据进行特征提取，分析客户基本信息，分期信息与交易信息。通过绘制各数据信息饼图和堆叠条形图来可视化不同用户间的特征，挖掘分期客户与非分期客户间的差异。认为分期客户较未分期客户更加年轻，经济能力较差但消费能力强，账单分期更加频繁，因此该类用户多会办理交易分期业务。

针对问题二，根据题目要求，首先分析客户的基本信息和交易特征，选取相关变量。接着我们使用 **SMOTE 算法**处理分期客户与未分期客户不平衡的问题。然后我们通过 **PCA** 对数据进行降维，将降维后的主成分输入**随机森林分类器**中建立**基于 PCA 与随机森林的客户分期意愿预测模型**。该预测模型的十折验证测试集的准确率分别为 **0.8463** 和 **0.7821**,MSE 分别为 **0.1537，0.2179**. 最后我们将 6 位客户的信息通过 PCA 降维后输入模型，得到他们的分期意愿评分分别为：**0.125**、**0.16**、**0.165**、**0.85**、**0.935**、**0.895**。

针对问题三，首先我们就客户的综合价值进行分析，j 建立**熵权法—TOPSIS 模型**，选择评估客户价值的关键因素，消费额度、信用记录、额度使用率、消费次数、负债率和对银行的贡献度六大因素。通过 **PCA** 在各因素内对数据进行降维，接着使用**熵权法**确定各指标的权重为：**0.2071，0.2890，0.0938，0.2394，0.0313，0.1395**，然后，运用 **TOPSIS** 方法对客户进行综合评分，通过计算每个客户与理想客户的接近度，评估其综合价值，得分越高即为综合价值越高的“黄金客户”。通过分析黄金客户的特征，我们发现银行的黄金客户为消费能力强，分期需求强，经济能力强且信用较好的用户。

针对问题四，我们基于传统的 **RFM 模型**引入新的负相关维度 NR，建立**基于组合赋权法的 RFMN 模型**进行信用卡客户的分类与价值分析。首先我们对传统 RFM 模型的维度进行重定义，并基于数据引入与目标值负相关的维度 NR，针对给出的高维数据按进行分类提取特征，并设置分箱，对数据进行赋值，再利用**层次分析法**与**熵权法**确定**组合赋权重**得到维度得分，最后构建 **K-Means 聚类模型**，利用 **Elbow Method** 确定最优聚类数量，基于维度得分将信用卡客户划分为**六个类别**，并基于分类分别给出营销策略与信用额度。据此 6 为客户的信用额度调整为：**60000**、**80000**、**45000**、**20000**、**30000**、**65000**。

关键词:　**PCA 随机森林 熵权法—TOPSIS 组合赋权法 K-Means**

一　问题的背景和重述

1.1　问题背景

信用卡作为便捷支付工具，随着经济和科技的发展，市场迅速扩展。然而，近年来信用卡市场增速放缓。根据《中国商业银行信用卡业务数字化专题研究 2023H1》报告，2022 年末全国信用卡在用发卡量为 7.98 亿张，同比下降 0.28%；2023 年这一数字下降至 7.67 亿张，同比下降 3.88%。信用卡市场趋于饱和，新发卡量增速持续下滑，获客难度加大，传统获客和促活方式不再适用。留住存量客户和利用大数据进行精准营销，激发消费潜力，是信用卡业务未来发展的关键问题。

1.2　问题重述

本文基于附件所提供的客户的基本信息及历史交易数据等数据，旨在通过建立相应模型解决以下四个问题：

1. 分析客户的基本信息和交易特征，比较分期客户与非分期客户之间的差异。

2. 构建客户的分期意愿评分模型，并设计一个测试方案验证评分效果。特别地，为指定的客户给出其分期意愿得分。

3. 构建客户综合价值评估模型，判断谁是该银行信用卡的“黄金客户”。

4. 选择合适的角度对客户进一步细分，制定针对不同客户群的营销策略以最大化银行收入，并调整指定客户的信用额度。

二　问题分析

2.1　问题一的分析

针对问题一，我们需要分析客户的基本信息和交易特征，比较分期客户与非分期客户之间的差异。首先，通过年龄分布、负债率、客户信用来分析客户基本信息。接着通过客户账单分期次数及金额、分期额度使用率、上一次账单分期距今时间来分析客户账单分期的信息，。其次，通过累计消费、消费次数来分析客户的交易特征。通过绘制各数据信息饼图和堆叠条形图来可视化不同用户间的特征，挖掘分期用户与非分期用户间的差异。

2.2　问题二的分析

针对问题二，我们需要构建客户的分期意愿评分模型，并设计一个测试方案验证评分效果。首先，分析客户的基本信息和交易特征，选取相关变量，如年龄、婚姻状况、账龄、累计消费、消费次数和信用额度等，作为模型的输入特征。然后，使用 SMOTE 技术处理数据不平衡问题，通过 PCA 对数据进行降维，保留主要特征，减少模型复杂度。接下来，使用随机森林分类器建立预测模型，通过模型训练得到客户的分期意愿评分。接着我们将 6 位用户的数据通过 PCA 降维后输入模型，得到他们的分期意愿评分。最后，设计测试方案，通过交叉验证和测试集评估模型性能，确保模型的准确性和稳定性。

2.3　问题三的分析

针对问题三，我们需要构建客户综合价值评估模型，确定谁是银行的“黄金客户”。首先，选择评估客户价值的关键因素，如消费额度、信用记录、额度使用率、消费次数、负债率和对银行的贡献度等。并且通过 PCA 在各因素内对数据进行降维，接着使用熵权法确定各指标的权重，确保评价的客观性和

准确性。然后，运用 TOPSIS 方法对客户进行综合评分，通过计算每个客户与理想客户的接近度，评估其综合价值。最后，按照得分对客户进行排序，得分越高的客户即为综合价值越高的“黄金客户”。

2.4 问题四的分析

针对问题四，我们采用组合赋权法的 RFMN 模型对客户进行细分，并制定针对不同客户群的营销策略，以最大化银行收入，并调整指定客户的信用额度。首先，构建 RFMN 模型，计算最近消费、消费频率和消费金额三个指标。通过层次分析法和熵权法相结合的方式进行赋权，得到综合权重。使用聚类算法，根据 RFMN 得分将客户分组，并通过肘部法确定最优聚类数。最后，针对不同客户群体制定营销策略，并调整其信用额度。

三 模型假设

1. 假设客户分期意愿仅与客户的基本信息及历史交易数据有关，与其余的主客观因素无关。
2. 假设若客户从未分期，则上一次账单分期距离现在天数为 9999。
3. 假设若客户为被银行拉黑的高风险客户，其分期额度为 1。

四 符号说明

符号	含义
D_i	距离负理想解的距离
D_i^+	距离正理想解的距离
C_i	综合得分
X_{ij}	第 i 个样本的第 j 个原始数据
X'_{ij}	第 i 个样本的第 j 个标准化处理后的数据
P_{ij}	第 i 个样本在第 j 个指标上的比重
e_j	第 j 个指标的信息熵
w_j	第 j 个指标的权重
Z^+	理想解
Z^-	负理想解
Z'_{ij}	第 i 个样本的第 j 个加权标准化值
k	数据集中样本的总数量
n	指标的总数量
α	标准化常数，$\alpha = \frac{1}{\ln k}$
w'_j	第 j 个维度的组合赋权权重
b_i	规范化后的第 i 个用户的第 j 个指标值

五　模型的建立与求解

5.1　问题一模型的建立与求解

5.1.1　客户基本信息的比较

对于数据中所给的婚姻状况、年龄、额度、账龄等数据进行分析，首先，我们对分期客户与非分期客户的比例进行比较，发现在 10000 例数据中分期客户占 16.49%，非分期客户占 83.51%，结果如下：

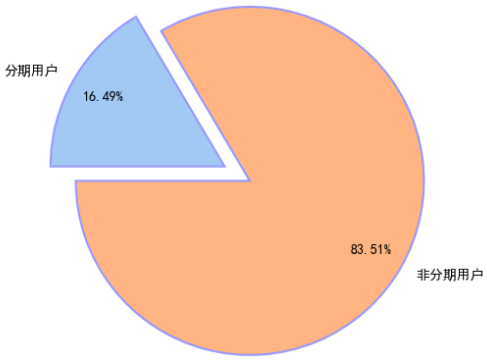

图 1: 分期客户与非分期客户的比例

针对分期客户与非分期客户之间的基本信息的差异，选取了以下几类数据特征进行比对：

1. 年龄分布与平均年龄

通过对信用卡分期用户与未分期用户的年龄分布分析，我们发现，18-25 岁的年轻用户和 45-55 岁的中年用户更倾向于选择分期付款，而 65 岁及以上的老年用户更倾向于不分期。结果如下：

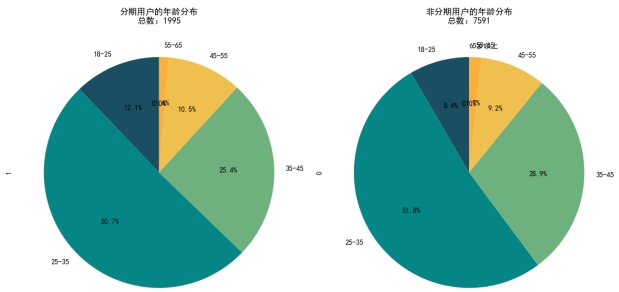

图 2: 分期客户与非分期客户年龄分布

通过对信用卡分期用户与未分期用户的年龄分布分析，我们发现分期用户的年龄中位数较低，18-45 岁的用户分期的比重更大，而未分期用户的年龄中位数约为 35 岁，较为老龄化。堆叠条形图显示，65 岁以上的用户多数未选择分期，而 18-25 岁的分期用户比例最高。可以认为年龄越低用户分期的比重更大。结果如下：

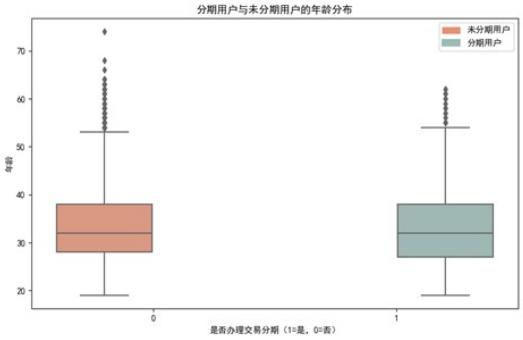

图 3: 分期客户与非分期客户年龄分布箱线图

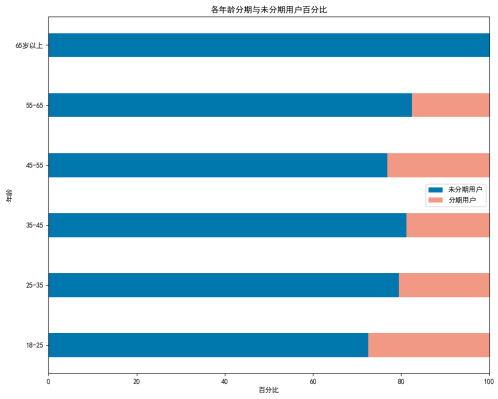

图 4: 各年龄段分期客户与非分期客户百分比

2. 负债率

通过对分期客户与未分期客户的负债率箱线图分析，我们发现分期客户中的负债率整体及中位数都较未分期用户高。但是高负债率用户以未分期用户为主，此类用户可能是因欠债而被限制分期。

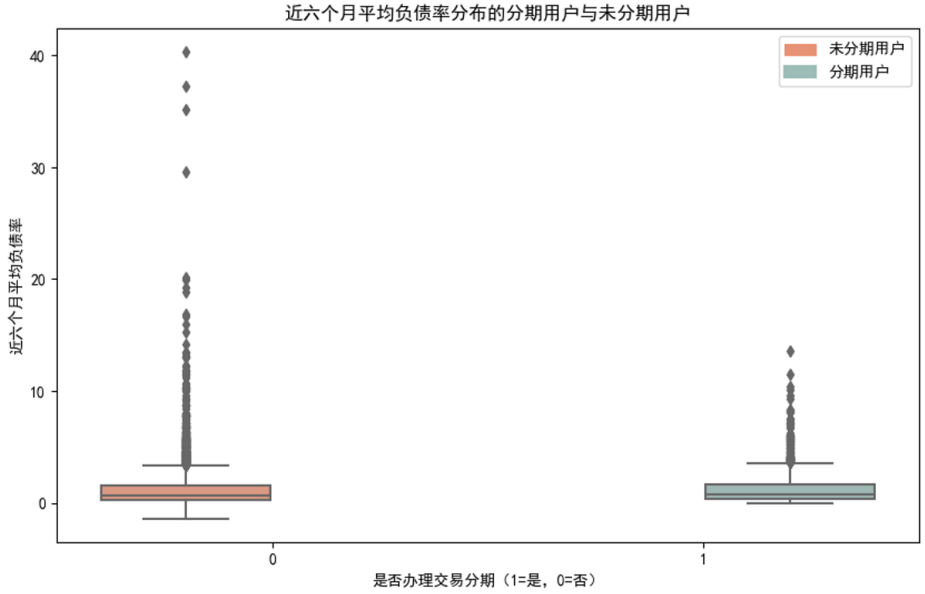

图 5: 分期客户与非分期客户负债率箱线图

3. 用户信用

通过对近六个月还款金额为最低还款的次数分析可知，未分期用户近六个月还款金额为最低还款的次数为 0 次居多，说明他们的付款能力较强。而分期用户近六个月还款金额为最低还款的次数偏高，说明很多分期用户的还款能力较差。

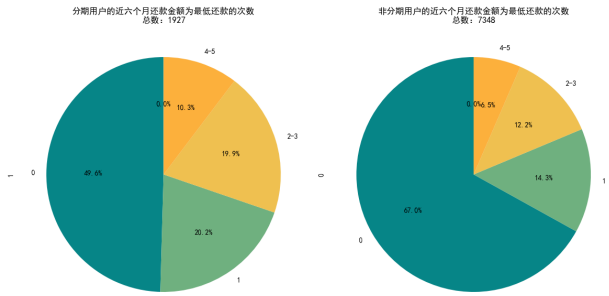

图 6: 分期客户与非分期客户近六个月还款金额为最低还款的次数

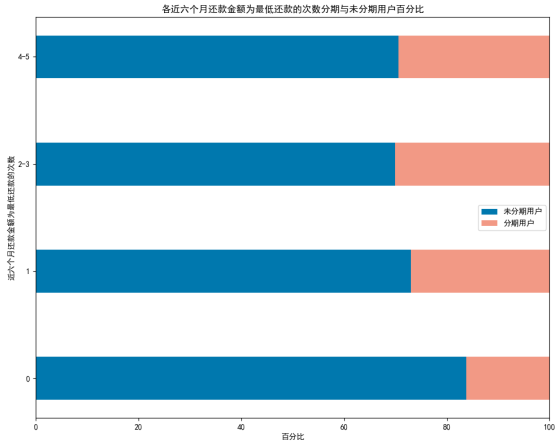

图 7: 分期客户与非分期客户近六个月还款金额为最低还款的次数百分比

综上所述，我们总结了三点分期客户与非分期客户之间的在基本信息上的差异，如下：

- 分期用户主要集中在 18-55 岁年龄段，而未分期用户主要集中在 65 岁及以上。分期用户的年龄偏低。
- 分期客户的整体负债率较未分期用户高，但是高负债率的用户中未分期用户更多

5.1.2 客户分期信息的比较

针对客户的账单分期次数，我们选取了以下几类数据进行比对：

1. 近六个月账单分期次数及金额

通过近六个月账单分期次数分析，未分期用户近期几乎没有进行账单分期，而分期用户近六个月分期次数更高。从堆叠条形图显示，尽管分期用户数量较少，账单分期次数与分期金额中分期用户更高。表明分期用户更愿意进行账单分期。

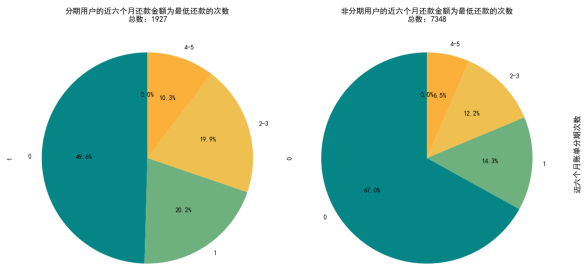

图 8: 分期客户与非分期客户近六个月还款金额为最低还款的次数

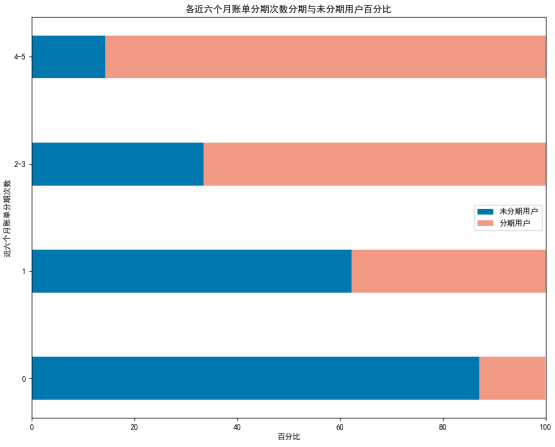

图 9: 各近六个月账单分期次数分期与未分期用户百分比

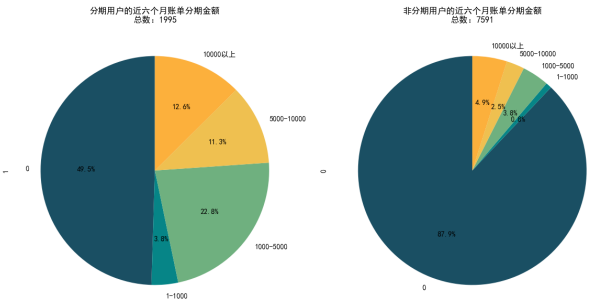

图 10: 分期客户与非分期客户近六个月账单分期金额

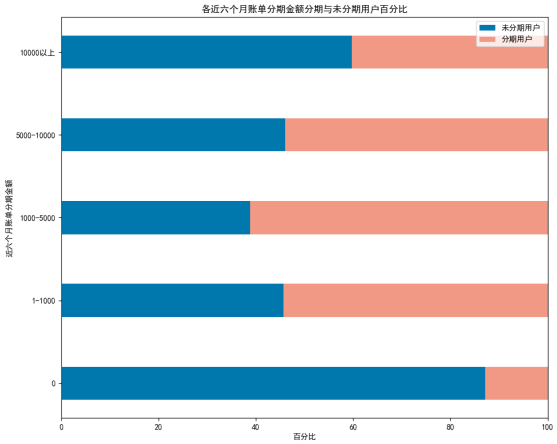

图 11: 各近六个月账单分期金额百分比

2. 本期分期额度使用率

通过对用户本期分期额度使用率分析，未分期用户本期额度使用率在 0-0.1 的占比很高（占比 26.7%），其次是 0.1-0.3（占比 19.2%）表明未分期用户的本期分期额度使用率偏低。而分期用户的在 0-0.1 的低频额度使用用户仅占比 8.5%，并且堆叠条形图显示，未分期用户在低频额度使用率中占比更高，而较高的额度使用率中，分期用户占比更高。表明未分期用户使用额度的意愿较低，而分期用户使用额度的意愿更高。

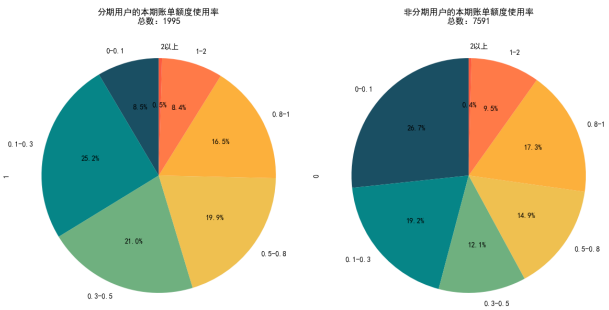

图 12: 分期客户与非分期客户本期账单使用率

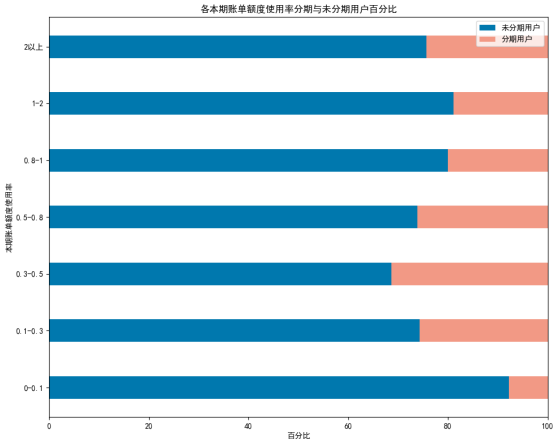

图 13: 各账龄状况下分期客户与非分期客户本期账单使用率百分比

3. 上一次账单分期距离现在天数

通过对用户上一次账单分期距离现在天数分析，未分期用户中从未分期的占比很高（占比 71.2%），其次是 180 天以上没有分期（占比 16.1%），可以认为绝大多数未开通分期的用户近期都没有进行账单分期。而分期用户中近半年有过账单分期的用户超过 50%，并且堆叠条形图显示，半年内有过账单分期的用户中分期用户占比较高，表明近期有过账单分期的用户开通交易分期的可能性更大。

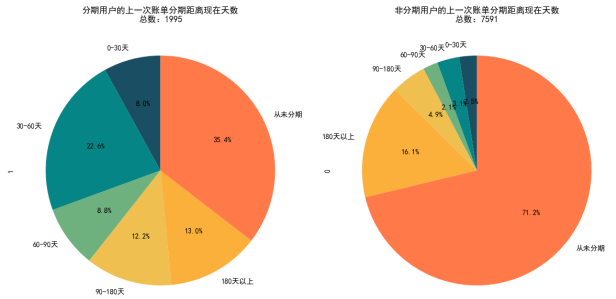

图 14: 分期客户与非分期客户上一次账单分期距离现在天数

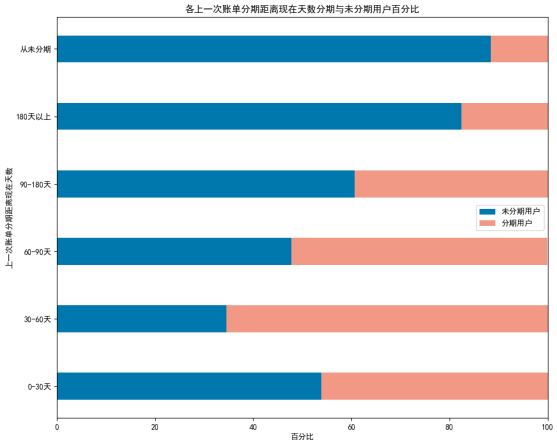

图 15: 分期客户与非分期客户上一次账单分期距离现在天数百分比

综上所述，我们总结了三点客户分期信息的比较，如下：

- 分期用户在近六个月的账单分期次数和金额明显高于未分期用户，表明他们更倾向于进行账单分期。
- 分期用户的分期额度使用率较高，特别是在较高额度使用区间中，分期用户占比显著高于未分期用户。
- 分期用户中，超过半数在最近半年内有账单分期记录，说明有分期经历的用户更倾向于再次分期。

5.1.3 客户交易特征的比较

针对分期客户与非分期客户之间的交易特征的差异，我们选取了以下几类数据进行比对：

1. 累计消费

通过对分期用户与未分期用户的消费累计分布进行分析，我们发现分期用户的消费集中在 500-2000 元（占比 18.8%）、2000-5000 元（占比 22.7%）、5000-10000 元（占比 19.3%）和 10000-30000 元（占比 25.0%）四个区间；未分期用户的消费则在 0-500 元（占比 22.2%、10000-30000 元（占比 22.1%）。这表明分期用户更多集中在中等消费区间，而未分期用户在高消费区间及低消费区间的比例更高，显示出未分期用户更倾向于一次性支付大额消费或者完全不消费。堆叠条形图进一步展示了各消费累计段的分期与未分期用户比例，发现除消费金额在 0-500 中未分期用户占比较多以外，其余各消费段虽分期用户总数较低，但占比较高，表明分期用户整体消费水平较高。结果如下：

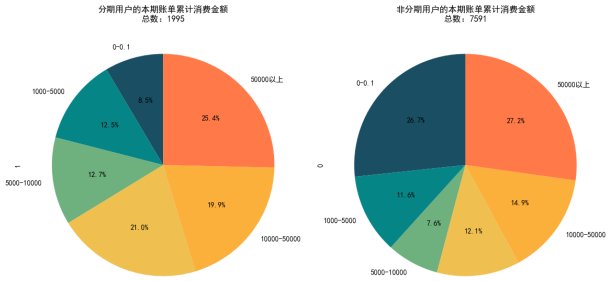

图 16: 分期用户与未分期用户本期账单累计消费

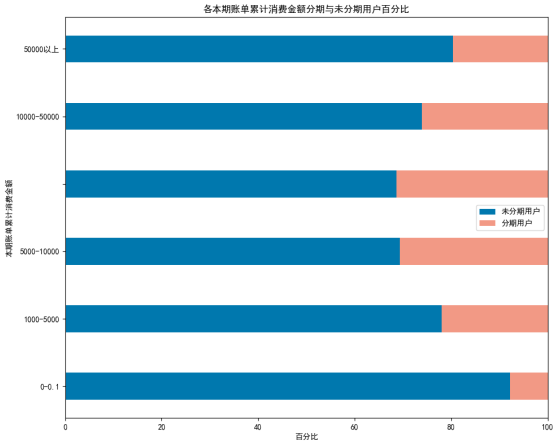

图 17: 分期用户与未分期用户本期账单累计消费百分比

2. 消费次数

通过对分期用户与未分期用户的消费次数分布进行分析，我们发现分期用户的消费主要集中在 2-5（占比 33.1%）、5-10 次（占比 19.6%）和 10-20 次（占比 12.7%）等区间；未分期用户的消费主要集中在 0-1 次（占比 15.8%）、1-2 次（占比 20.0%）和 2-5 次（占比 27.5%）等区间。堆叠条形图显示，分期用户在较高消费次数区间（200 次以上）的比例较高，而未分期用户在较低消费次数区间（0-1 次）的比例更高。这表明分期用户的消费频率较高，而未分期用户的消费频率较低。结果如下：

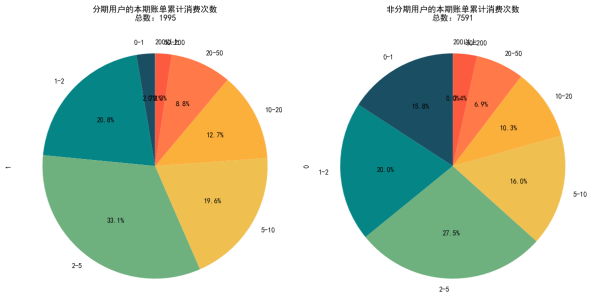

图 18: 分期用户与未分期用户的消费次数

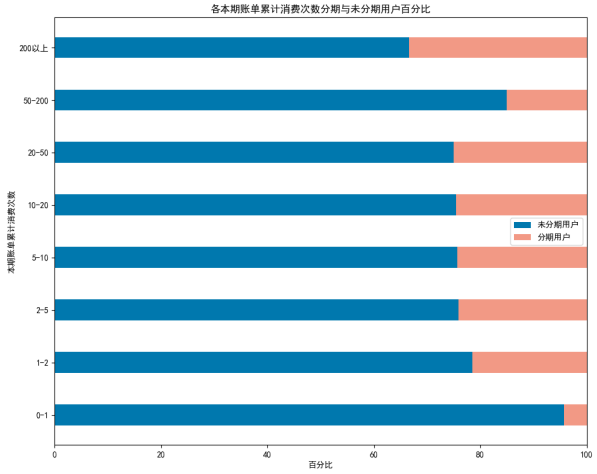

图 19: 各本期账单累计消费次数段的分期用户与未分期用户百分比

综上所述，我们总结了两点分期客户与非分期客户之间的在交易特征上的差异，如下：

- 分期用户的消费主要集中在低消费和中等消费区间，而未分期用户在高消费区间的比例更高，未分期用户更倾向于一次性支付大额消费。
- 分期用户的消费频率较高，尤其是在 200 次以上的消费次数区间；未分期用户的消费频率较低，主要集中在 0-1 次的消费次数区间。

5.2 问题二模型的建立与求解

为构建客户的分期意愿评分模型,并设计一个测试方案以验证评分的使用效果,我们通过 SMOTE[1] 进行过采样处理来处理不平衡数据，然后使用 PCA[2] 降维和随机森林 [3] 分类器建立了一个预测客户分期意愿的模型。此模型会输出用户分期的概率，以此作为分期意愿得分。

5.2.1 SMOTE 过采样

由于样本的分期人数偏少，影响预测模型的结果，故采用 SMOTE 过采样技术来处理不平衡数据。SMOTE 是基于随机过采样算法的一种改进方案，由于随机过采样采取简单复制样本的策略来增加少数类样本，这样容易产生模型过拟合的问题，即使得模型学习到的信息过于特别而不够泛化。算法流程如下：

1. 对于少数类中每一个样本 x，以欧氏距离为标准计算它到少数类样本集中所有样本的距离，得到其 k 近邻。

2. 根据样本不平衡比例设置一个采样比例以确定采样倍率 N，对于每一个少数类样本 x，从其 k 近邻中随机选择若干个样本，假设选择的近邻为 x_n。

3. 对于每一个随机选出的近邻 x_n，分别与原样本按照如下的公式构建新的样本：

$$\mathbf{x}_{\text{new}} = \mathbf{x} + \text{rand}(0,1) \times (\tilde{\mathbf{x}} - \mathbf{x})$$

其中，$\mathbf{x}_{\text{new}}$ 为新生成的样本，$\mathbf{x}$ 为原始少数类样本，$\tilde{\mathbf{x}}$ 为从 k 近邻中随机选择的一个样本，$\text{rand}(0,1)$ 为取值在 0 到 1 之间的随机数。

5.2.2 基于 PCA 与随机森林的客户分期意愿预测模型

在建立客户分期意愿预测模型时，我们先利用 PCA 对数据进行特征缩放及降维，具体步骤如下：

1. 标准化处理：

使用 Min-Max Scaler，通过线性变换将原始数据缩放到指定的最小值和最大值之间。

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{1}$$

2. 数据中心化：

假设数据 $\{a^{(1)}, a^{(2)}, \ldots, a^{(m)}\}$ 已中心化，且投影后的新坐标系为 $\{w_1, w_2, \ldots, w_n\}$。

3. 低维投影：

数据在低维空间中的坐标为

$$z_i^{(j)} = w_i^T a^{(j)} \tag{2}$$

样本点 $a^{(j)}$ 的低维坐标为

$$z^{(j)} = (z_1^{(j)}, z_2^{(j)}, \ldots, z_n^{(j)})^T \tag{3}$$

4. 最大化投影方差：

投影方差为

$$\frac{1}{m}\sum_{i=1}^{m}(w^T a^{(i)})^2 \tag{4}$$

最大化投影方差等价于

$$\arg\max_{w} w^T X^T X w \quad \text{s.t.} \quad w^T w = I \tag{5}$$

利用拉格朗日目标函数

$$J(W) = \mathrm{tr}(W^T X^T X W) + \lambda(W^T W - I) \tag{6}$$

对 W 求导得

$$X^T X W = (-\lambda)W \tag{7}$$

其中，W 为 $X^T X$ 的特征向量，$-\lambda$ 为特征值。将前 k 个特征向量作为列组成矩阵 W，XW 表示原始数据投影到新的 k 维空间。

在对原始高维数据降维后，我们使用随机森林分类器进行模型训练。随机森林作为一种基于决策树的机器学习算法，主要用于分类和回归任务。它由多个决策树组成，每个决策树都是独立训练的。其核心思想是通过组合多个决策树的预测结果，来提高模型的准确性和稳定性。随机森林模型的建模过程主要包括以下步骤：

1. 自助采样：

从原始数据集中使用自助采样法抽取样本，形成多个子数据集。

2. 构建决策树：

对每个子数据集，构建一个决策树。在每个节点，随机选择一部分特征进行分裂。当每个样本有 M 个属性时，在决策树的每个节点需要分裂时，随机从这 M 个属性中选取出 m 个属性，满足条件 $m \ll M$。然后从这 m 个属性中采用某种策略来选择 1 个属性作为该节点的分裂属性。

3. 重复分裂：

决策树形成过程中每个节点都要按照上述方式分裂，直到不能再分裂为止。注意整个决策树形成过程中没有进行剪枝。

4. 生成随机森林：

重复上述步骤，直到生成指定数量的决策树，这样就构成了随机森林。

最终的预测模型为：

$$\hat{P}(y = 1 \mid X_{\mathrm{PCA}}) = \frac{1}{N}\sum_{i=1}^{N} T_i(X_{\mathrm{PCA}}) \tag{8}$$

其中，T_i 是第 i 棵决策树，N 是决策树的数量，$\hat{P}(y = 1 \mid X_{\mathrm{PCA}})$ 是预测的分期意愿概率。

5.2.3 模型求解

对于附件中的数据集，我们将模型视为一个二分类模型，结果大于 0.5 则表示用户会分期，结果小于 0.5 则表示用户不会分期。首先，我们将数据集按 90:10 的比例划分为训练集和验证集，保证验证集中 0 和 1 的比例与训练集一致。接下来，我们对训练集中不平衡的数据进行 SMOTE 过采样，以平衡正负样本比例。然后，我们对数据进行特征缩放，并使用主成分分析（PCA）进行降维，以减少数据维

度，降低模型复杂度，同时保留尽可能多的原始信息。在处理后的数据上，我们使用随机森林模型进行训练，并通过网格搜索优化参数，最终得到最优参数，结果如下：

针对问题表格中的客户，我们给出其分期意愿得分，结果如下：

表 1: 模型最优参数

PCA 维度	树的数量	最大深度
15	200	30

同时，我们设计了对应的测试方案：将用户数据使用 PCA 降维后输入模型，得到预测的是否分期的概率，接近于 0 表示用户分期的可能性很小，接近于 1 表示用户分期的可能性很大。以 id 为 211 的客户为例，其分期意愿得分计算过程如下所示：

1. **输入数据处理：**客户的数据经过特征缩放和 PCA 降维处理，得到降维后的特征向量 X_{PCA}。特征缩放使用的是标准化方法，将各特征值转换为均值为 0、标准差为 1 的标准正态分布。接着，利用主成分分析（PCA）将高维特征数据投影到较低维度的空间中，从而降低数据的维度，减少噪声和冗余信息。

2. **模型预测：**将处理后的特征向量 X_{PCA} 输入到训练好的随机森林模型中。随机森林模型由 200 棵决策树组成，每棵决策树会基于这些输入数据独立做出判断，输出该客户选择分期的概率值。

3. **概率平均：**随机森林模型中的 200 棵决策树分别给出了该客户选择分期的概率值，最终的分期意愿得分是所有决策树输出结果的平均值，即：

$$\hat{P}(y = 1 \mid X_{\text{PCA}}) = \frac{1}{200} \sum_{i=1}^{200} T_i(X_{\text{PCA}}) \tag{9}$$

其中，$T_i(X_{\text{PCA}})$ 表示第 i 棵决策树预测该客户选择分期的概率。对于该客户，模型综合所有决策树的结果后，计算得出的分期意愿得分为 0.125。

表 2: 客户分期意愿得分

id	211	3047	5971	7090	8576	9982
客户原始意向	0	0	0	1	1	1
分期意愿得分	0.125	0.16	0.165	0.85	0.935	0.895

5.2.4 模型检验

为了评估模型的性能，我们进行了十折交叉验证和测试集评估。在十折交叉验证中，模型的平均准确率为 0.8463，标准误差为 ±0.0339，均方误差（MSE）为 0.1537，标准误差为 ±0.0339。这表明模型在训练集上的表现较为稳定，具有较好的泛化能力。在测试集上的表现同样令人满意，准确率达到了 0.7821，MSE 为 0.2179。这些结果表明，尽管模型在测试集上的表现略低于交叉验证的结果，但整体上依然具有较高的预测准确性。此外，我们对特定客户进行了分期意愿预测，预测概率与原始意愿的比较

进一步验证了模型的有效性。具体预测结果表明，模型能够较准确地捕捉用户的分期意愿，为实际应用提供了有力支持。

表 3: 十折交叉验证和测试集结果

	准确率	标准误差	MSE
十折交叉验证	0.8463	±0.0339	0.1537 ±0.0339
测试集	0.7821	-	0.2179

5.3 问题三模型的建立与求解

5.3.1 模型求解

问题三要求构建客户综合价值评估模型，对客户的综合价值进行量化分析，从而确定银行信用卡的“黄金客户”。因此，我们建立熵权法——TOPSIS 模型 [4]，选取消费额度、客户信用、额度使用率、消费次数、负债率和银行贡献度这六个因素，其中各因素因子如下图，使用 PCA 降维保留主成分来综合评估客户的综合价值。

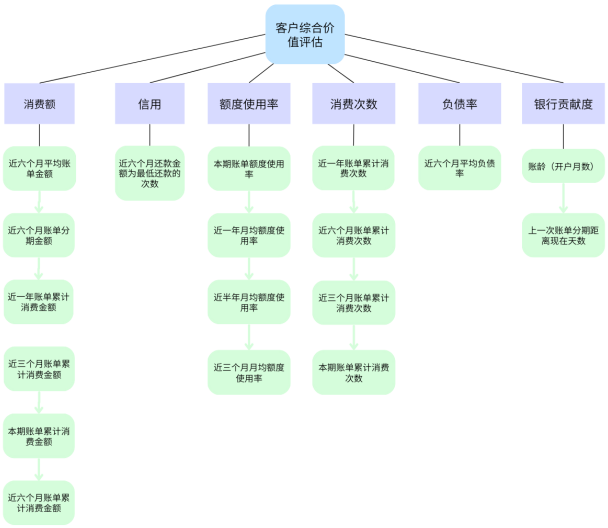

图 20: 各因素的因子

5.3.2 熵权法—TOPSIS 模型

熵权法-TOPSIS 模型是一种将根据各指标所含信息的差异性进行各指标客观赋权和逼近于理想解的排序结合起来的一种评价模型。其具体步骤如下：

1. 数据处理：

首先对原始数据进行标准化处理，以消除不同指标间的量纲影响，使数据具有可比性。对于正向指标和负向指标分别进行如下转换：

对正向指标：

$$X'_{ij} = \frac{x_{ij} - \min(X_{ij})}{\max(X_{ij}) - \min(X_{ij})} + 0.0001 \tag{10}$$

对负向指标：

$$X'_{ij} = \frac{\max(X_{ij}) - x_{ij}}{\max(X_{ij}) - \min(X_{ij})} + 0.0001 \tag{11}$$

2. 熵值法计算权重：

计算每个指标在样本中的比重：

$$P_{ij} = \frac{X'_{ij}}{\sum_{i=1}^{n} X'_{ij}} \tag{12}$$

根据比重矩阵计算熵值，熵值是用来表示信息不确定性的一种度量：

$$e_j = -k \sum_{i=1}^{n} P_{ij} \ln P_{ij}, \quad k = \frac{1}{\ln n} \tag{13}$$

再通过熵值计算权重，权重反映了每个指标在综合评价中的重要性：

$$w_j = \frac{1 - e_j}{\sum_{j=1}^{m}(1 - e_j)} \tag{14}$$

3.TOPSIS 计算：

将标准化后的数据乘以熵值法计算出的权重，得到加权标准化矩阵：

$$Z'_{ij} = Z_{ij} \cdot w_j \tag{15}$$

确定理想解和负理想解。理想解是各指标的最优值，负理想解是各指标的最差值：

理想解：

$$Z^+ = (\max(Z'_{ij})), \quad j = 1, 2, \ldots, m \tag{16}$$

负理想解：

$$Z^- = (\min(Z'_{ij})), \quad j = 1, 2, \ldots, m \tag{17}$$

计算每个样本距离理想解和负理想解的距离：

距离理想解的距离：

$$D_i^+ = \sqrt{\sum_{j=1}^{m}(Z'_{ij} - Z^+)^2} \tag{18}$$

距离负理想解的距离：

$$D_i^- = \sqrt{\sum_{j=1}^{m}(Z'_{ij} - Z^-)^2} \tag{19}$$

4. 综合评价：

计算各评价对象相对接近理想解的程度，综合得分越高，表示该对象越接近理想解：

$$C_i = \frac{D_i^-}{D_i^+ + D_i^-} \tag{20}$$

最后，按 C_i 大小排序，结合综合评价指数进行分析。综合得分 C_i 越接近 1，表示该对象的评价越优。

5.3.3 模型求解

将各特征分为 6 大因素，用于评估客户的综合价值。每个因素包含多个特征，通过 PCA 降维发现因素 2 与因素 5 只有一个特征，因此不做主成分分析，PCA 降维结果如下：

表 4: 各因子的解释方差比及成分矩阵

因子	解释方差比	成分 1	成分 2	成分 3	成分 4	成分 5	成分 6
Factor 1	0.7177	0.4470	0.4929	0.4232	0.4180	0.4336	0.1238
Factor 3	0.8565	0.6528	0.3653	0.3820	0.5426	-	-
Factor 4	0.8868	-0.0332	-0.0269	-0.0195	-0.0093	-0.9989	-
Factor 6	0.5232	0.8362	0.5485	-	-	-	-

依据熵权法-TOPSIS 模型利用 Python 确定评价指标的权重, 权重如下所示:

表 5: 各个指标的权重

指标	factor1	factor2	factor3	factor4	factor5	factor6
权重	0.2071	0.2890	0.0938	0.2394	0.0313	0.1395

完整公式如下：

$$\text{value}_j = r_{j1} \times 0.2071 + r_{j2} \times 0.2890 + r_{j3} \times 0.0938 + r_{j4} \times 0.2394 + r_{j5} \times 0.0313 + r_{j6} \times 0.1395 \quad (21)$$

最后，我们得出所有客户的综合得分，分布如下：

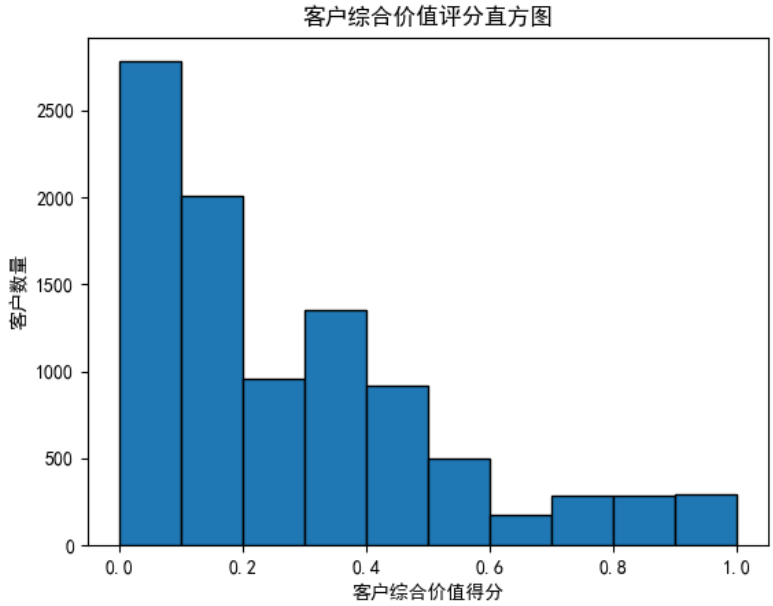

图 21: 客户综合价值评分直方图

同时，我们统计了各类用户的百分比，将客户分为三类：

- 黄金客户：比分区间 0.8-1.0
- 白银用户：比分区间 0.5-0.8
- 普通用户：比分区间 0.0-0.5

结果如下：

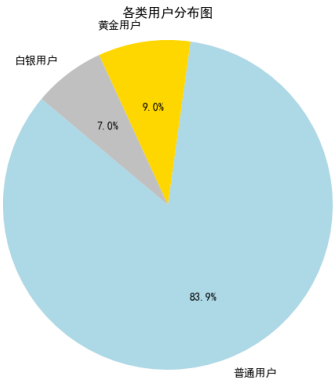

图 22: 各类用户分布图

绝大部分用户是普通用户，黄金客户和白银用户占比相对较小。这说明高价值用户（如黄金用户和白银用户）在总用户中所占比例较低。

此外，我们对黄金用户的相关特征进行分析，如下图：

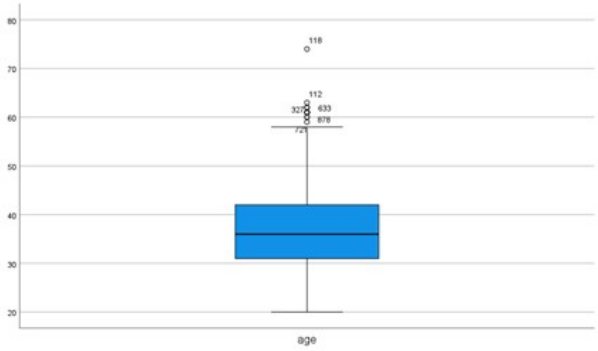

图 23: 黄金客户年龄箱线图

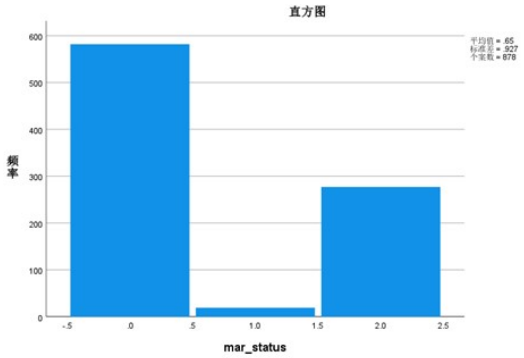

图 24: 黄金客户婚姻状况直方图

通过箱线图分析，银行信用卡的主要用户年龄集中在 32 到 43 岁之间，这一年龄段的用户大多数为中年人，是高净值用户的重要组成部分。从婚姻状况来看，未婚和婚姻状况未知的用户占比最大，这类用户没有家庭负担，消费自由度较高，更倾向于大额消费和分期付款。

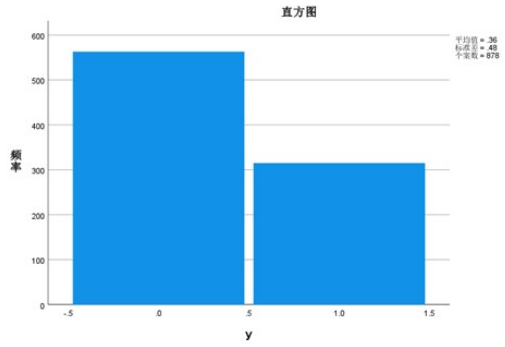

图 25: 分期意向直方图

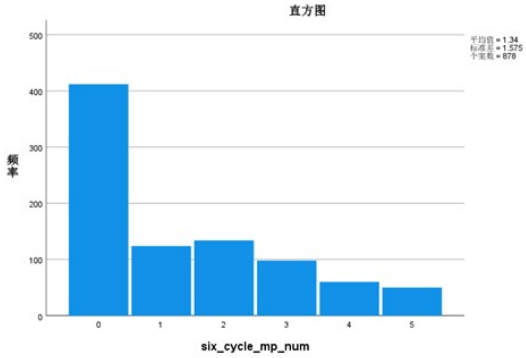

图 26: 6 个月分期次数直方图

通过上图可以看出，银行信用卡的分期用户数量明显高于普通用户，且在过去 6 个月内分期次数较多。

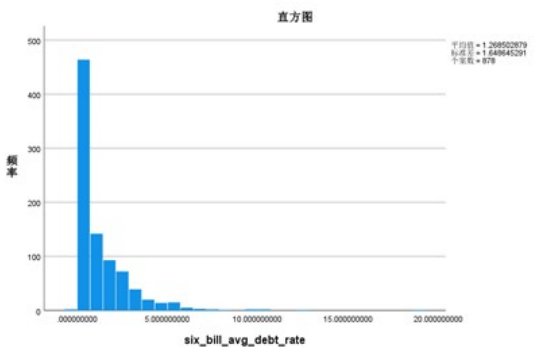

图 27: 平均负债率直方图

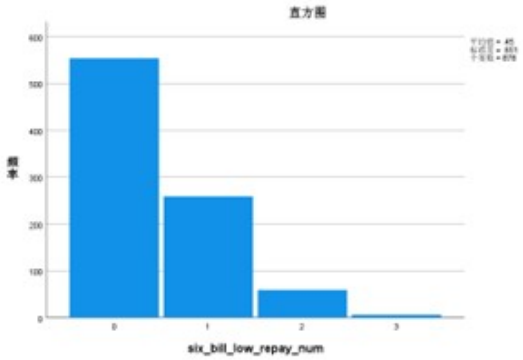

图 28: 还款金额为最低金额次数直方图

通过上图可以看出，用户的平均负债率集中在 0 到 2 之间，说明大部分用户的负债率较低，平均值为 1.29，标准差为 1.68。大多数用户在过去六个月中没有低还款记录，平均低还款次数为 0.45，标准差为 0.65，显示用户按时还款情况良好。

表 6: RFM 指标均值

指标	描述	均值
this_bill_rate	本次账单比率	0.2793
consume_amt_session	消费金额（过去 1 次）	9123.0118
consume_amt_session3	消费金额（过去 3 次）	29025.3821
six_bill_num	6 次账单数量	5.94
six_cycle_mp_num	6 个周期消费数量	1.34

根据对上表各类指标的统计，我们得出“黄金客户”信用良好，近半年分期额度使用适中且按时还，近期累计消费次数与金额高，近半年账单次数与平均金额高，信用卡额度高，且近半年分期使用频繁且额度高。

5.4 问题四模型的建立与求解

5.4.1 模型建立

传统的 RFM 模型对于问题四要求具有一定的局限性，针对不同的客户群，银行将采用怎样的营销策略的要求，我们参考了传统的 RFM 模型与基于组合赋权法的 RFMN 模型的信用卡用户价值分析 [5], 并对其中的 RFMN 指标进行修改，引入新的负相关 NR 维度，通过层次分析法权重、熵权法权重、组合赋权权重确定权重，再构建 K-Means 聚类模型, 利用 Elbow Method 确定最优聚类数量，建立最优 K-Means 聚类模型, 确定每个用户的归类情况并给出相应的营销策略。

5.4.2 基于 PCA 的特征提取

RFM 模型是衡量客户价值和客户创利能力的重要工具和手段之一，有消费才有创利可能，因此它本身的 R、F、M 这三个维度的定义是围绕着客户的消费情况而确定的。本文基于获得的数据，为了使

得 RFM 模型能够更好地服务于本文的研究对象，对传统 RFM 模型 R、F、M 维度进行了重新定义，结果如下：

R(Recency) ：用户最近一次分期时间点与分析截止时间点间的间隔。

F(Frequency)：观测时间内的消费次数与分期次数。

M(Monetary) ：观测时间内的消费金额与额度使用。

NR(Number and Rate)：还款金额为最低还款的次数与负债率。

表 7: 传统 RFM 模型

R(Recency)	**F(Frequency)**	**M(Monetary)**
用户最近一次消费时间点与分析时间点间的间隔	观测时间内的消费次数	观测时间内的消费总金额

针对问题四给出的高纬度数据，数据间具有一定相关性，我们采用 PCA 降维法，对于每一 R、F、M、NR 维度选取与之相关的数据，并采用 PCA 降维得到每一维度的特征，每一维度选取的数据如下：

表 8: RFM 指标说明

指标	**描述**
R:	最后一次消费天数（last_mp_days）、账户年龄（xaccount_age）
F:	过去 12 次消费次数（consume_num_session12）、过去 6 次消费次数（consume_num_session6）、过去 3 次消费次数（consume_num_session3）、过去 1 次消费次数（consume_num_session）、6 次账单数量（six_bill_num）、6 个周期的消费数量（six_cycle_mp_num）、12 个月的 EPP 数量（epp_nbr_12m）
M:	6 次账单平均金额（six_bill_avg_amt）、过去 12 次消费金额（consume_amt_session12）、过去 6 次消费金额（consume_amt_session6）、过去 3 次消费金额（consume_amt_session3）、过去 1 次消费金额（consume_amt_session）、6 个周期的平均消费金额（six_cycle_mp_avg_amt）
N:	6 次账单低偿还次数（six_bill_low_repay_num）、6 次账单平均债务率（six_bill_avg_debt_rate）

5.4.3 基于组合赋权法的 RFMN 模型

对于银行客户的分类，我们建立基于组合赋权法的 RFMN 模型，采用五分法（0.2、0.4、0.6、0.8）对经过预处理后 R、F、M、NR 数据进行分箱处理，具体得分区间如下表所示：

表 9: RFMN 模型评分标准

分值	R	F	M	N
1	[-1.13, -1.00)	[-1.24, -0.84)	[-1.32, -1.09)	[-1.57, -0.78)
2	[-1.00, -0.79)	[-0.84, -0.66)	[-1.09, -0.84)	[-0.78, -0.48)
3	[-0.79, 0.47)	[-0.66, -0.41)	[-0.84, -0.38)	[-0.48, -0.07)
4	[0.47, 0.87)	[-0.41, 0.24)	[-0.38, 0.60)	[-0.07, 0.63)
5	[0.87, 4.18]	[0.24, 36.92]	[0.60, 62.49]	[0.63, 16.68]

通过上表的处理，即可计算得到各信用卡用户的 R、F、M、NR 的原始得分，但需要按以下规则进行原始得分设置

R: 最近分期日期越接近分析截止日期，得分越高；

F: 观测时间内消费次数越多，得分越高，最高 5 分，最低 1 分；

M: 观测时间内消费金额越高，得分越高，最高 5 分，最低 1 分；

N: 观测时间内还款金额为最低还款的次数越少，得分越高，最高 5 分，最低 1 分；

此外，主观赋权法和客观赋权法都曾被单独引用到信用卡风险模型中，用于确定影响因素的权重。主观赋权法侧重决策者的意图（即是决策者对不同指标的重视程度），主观性较强；而客观赋权法侧重客观，可能会出现权重和实际相反的情况。而将两者相结合的组合赋权法平衡了两者的优劣。组合赋权法可以弥补单一赋权带来的主观性/客观性过强的不足，同时又保留了主观随机性和客观公正性，使得权重能够实现主客观统一，评价真实公正。并且组合赋权法作为主客观统一的确定用户指标权重的方法，却还未被学者用于研究信用卡风险。因此本文使用组合赋权法确定 RFM 模型中 R、F、M 三个维度的权重并引入 NR 维度的权重，这里用到的主观赋权法为层次分析法，客观赋权法为熵权法。因此，我们利用组合赋权法对模型权重的计算，具体步骤如下：

1. 层次分析法计算维度权重

使用层次分析法获取三个维度各自的权重，首先需要按照层次分析法的原则，将相关因素分解成三层，得到基于 **RFMN** 的层次结构。

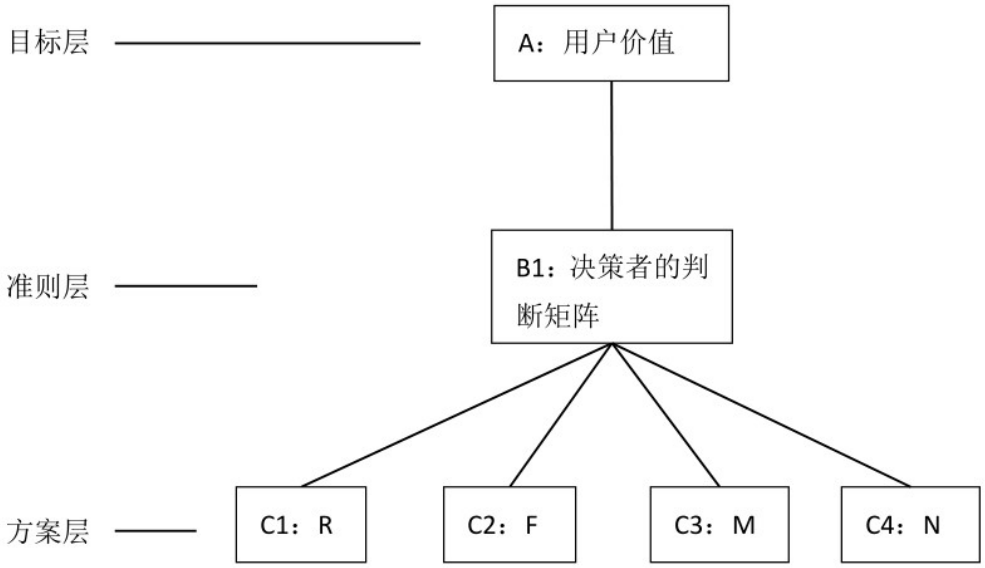

图 29: 基于 RFMN 的层次结构 [5]

将判断矩阵作为权重的数据来源根据，再进行层次分析，得到各维度的权重 W_j'。

2. 熵权法下的维度权重计算

记 **RFMN** 模型所有维度的原始得分数据集为 $\mathbf{OS} = (os_{ij})_{m\times n}$，其中 m 为客户总数，n 为维度个数，$os_{ij}(i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$ 为第 i 个客户第 j 个维度原始得分。

计算在第 j 个维度下，第 i 个客户的维度原始得分所占比重 p_{ij}：

$$p_{ij} = \frac{os_{ij}}{\sum_{i=1}^{m} os_{ij}}. \tag{22}$$

记第 j 个维度的信息熵为 E_j：

$$E_j = -\alpha \sum_{i=1}^{m} p_{ij} \ln p_{ij}, \text{其中} \alpha = \frac{1}{\ln m}. \tag{23}$$

因此，第 j 个维度的权重为 W_j''：

$$W_j'' = \frac{(1 - E_j)}{\sum_{j=1}^{n}(1 - E_j)}. \tag{24}$$

3. 组合赋权法

层次分析法作为主观赋权法侧重决策者的意图，主观性较强。而熵权法作为客观赋权法侧重客观，可能会出现权重和实际相反的情况。而将两者相结合的组合赋权法平衡了两者的优势。组合赋权法可以弥补单一赋权带来的主观性/客观性过强的不足，同时又保留了主观随机性和客观公正性，使得权重能够实现主客观统一，评价真实公正。

将由主观赋权法得到的属性权重记为 $\mathbf{W} = (W_1, W_2, \ldots, W_n)^T$，而由客观赋权法得到的属性权重记为 $\mathbf{W}' = (W_1', W_2', \ldots, W_n')^T$。两组权重向量分别满足：

$$0 \le W_j \le 1, \sum_{j=1}^{n} W_j = 1; \quad 0 \le W_j' \le 1, \sum_{j=1}^{n} W_j' = 1. \tag{25}$$

则组合赋权权重为：

$$\mathbf{W} = T\mathbf{W} + U\mathbf{W}'. \tag{26}$$

其中，T, U 分别为 $\mathbf{W}, \mathbf{W}'$ 的重要程度。

为了使得 $\mathbf{W} = T\mathbf{W} + U\mathbf{W}'$ 中的权重满足

$$0 \le W_j \le 1, \sum_{j=1}^{n} W_j = 1, \tag{27}$$

对 T, U 进行归一化处理，对其进行定义：

$$\bar{T} = \frac{\sum_{i=1}^{m} b_i W_j'}{\sum_{i=1}^{m}\sum_{j=1}^{n} b_i (W_j + W_j')}, \tag{28}$$

$$\bar{U} = \frac{\sum_{i=1}^{m} b_i W_j'}{\sum_{i=1}^{m}\sum_{j=1}^{n} b_i (W_j + W_j')}. \tag{29}$$

其中，b_j 为规范化后的第 i 位用户第 j 个指标值，m 为客户总数。

5.4.4 K-Means 聚类

K-Means 聚类算法 [6] 是经典的基于划分的聚类算法之一。算法根据预先设定的聚类数目和初始聚类中心，基于某种分配原则进行迭代划分，然后重复执行更新中心点和分配对象集的操作，直到每个对象被划分到最合适的簇类中去。最终目标是使得类内的数据点尽可能接近，而类间的数据点尽可能远离。即是使准则函数 F 达到最小，此时得到的聚类结果能够实现上述目的，使得聚类效果达到最佳。判断聚类效果优劣的准则函数 F 为：

$$\min F = \sum_{j=1}^{k} \sum_{i=1}^{n} \|x_i - c_j\|^2 \tag{30}$$

其中，k 为聚类数目，n 为数据点总数，x_i 为第 i 个数据点，c_j 为第 j 个聚类中心。

K-Means 聚类算法的具体步骤如下：

1. 确定聚类数目 k；
2. 确定初始聚类中心 $c_1, c_2, \ldots, c_k$；
3. 计算每个数据点 x_i 与所有聚类中心的欧式距离 d，将 x_i 分配到距离最小的聚类中心；
4. 重新计算每个聚类的均值，作为新的聚类中心；
5. 重复步骤 (3) 和 (4)，直到聚类中心不再发生变化，算法终止。

5.4.5 模型求解

根据根据基于组合权法得到的属性权重，使用 K-Means 聚类算法对模型结果进行聚类，并提供手肘法确定 K 值，将结果分为 6 类，分别为：重要挽留用户，重要保持用户，一般挽留用户，一般保持用户，重要发展用户和重要价值用户，结果如下：

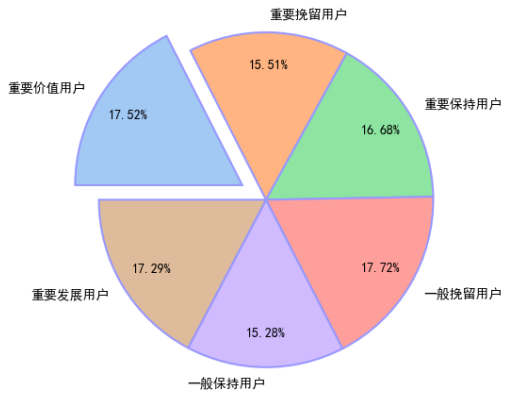

图 30: 各类用户分布比例

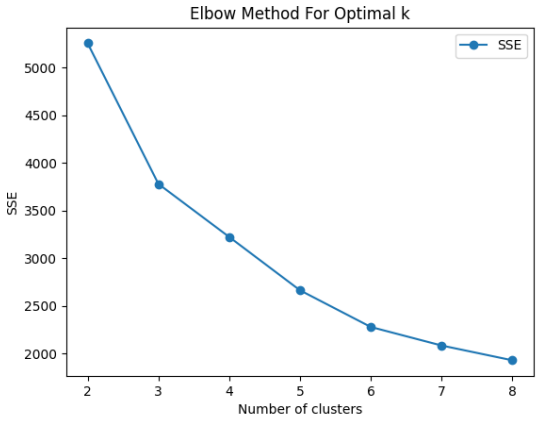

图 31: Elbow Method

根据上表我们可以得出：

表 10: 聚类结果

类别	中心值								样本个数	所占比例	用户定义
	RS	次序	FS	次序	MS	次序	NS	次序			
1	0.795	1	1.356	1	1.630	1	0.343	2	1679	17.52%	重要价值用户
2	0.449	5	0.521	5	1.244	3	0.356	3	1487	15.51%	重要挽留用户
3	0.624	3	0.897	3	1.575	2	0.316	1	1599	16.68%	重要保持用户
4	0.395	6	0.376	6	0.505	5	0.400	5	1699	17.72%	一般挽留用户
5	0.612	4	0.874	4	0.543	6	0.418	5	1465	15.28%	一般保持用户
6	0.679	2	1.278	2	0.866	4	0.459	6	1657	17.29%	重要发展用户

1. 第一类用户的 RS、FS、MS 中心值最大，NS 中心值第二，说明第一类用户最近有分期行为，并且历史支付金额大、次数多，负债率低信用度高，可将其定义为重要价值客户。

2. 第二类用户的 MS 和 NS 排名第三，但 FS、RS 中心值排名靠后，说明第二类用户历史交易金额较大且诚信度较好，但最近交易行为较少，可将其定义为重要挽留客户。

3. 第三类用户的 MS 中心值排名第二，NS 中心值排名第一，最近交易较为频繁，说明第三类用户是重要保持用户。

对于不同类型的客户群，我们制定了以下策略：

1. 重要挽留用户 (15.51%)

- **特征**：这些用户对银行有较高的贡献，但近期可能减少了使用频率或有离开的风险。
- **策略**：提供个性化的服务和优惠，例如专属客服、生日礼券、积分翻倍等。定期进行满意度调查，及时回应用户反馈，防止用户流失。

2. 重要保持用户 (16.68%)

- **特征**：这些用户对银行的忠诚度较高，消费频率和金额都处于较高水平。
- **策略**：重点维持，通过提升用户体验来增强其忠诚度。可以提供 VIP 专享服务、提升信用额度、邀请参与高端活动等，保持用户的高满意度和忠诚度。

3. 一般挽留用户 (17.72%)

- **特征**：这些用户的活跃度和贡献度一般，但有流失的风险。
- **策略**：通过发送关怀短信、优惠券等方式来激励其消费。可以推出限时优惠、打折活动，吸引用户重新活跃。

4. 一般保持用户 (15.28%)

- **特征**：这些用户的消费频率和金额一般，且对银行的依赖性较低。
- **策略**：通过增加附加价值来增强用户粘性。提供一些常规的优惠活动、提升用户体验，例如优化 APP 使用体验、提供更多的理财产品选择等。

5. 重要发展用户 (17.29%)

- **特征**：这些用户目前的贡献度较高，且有进一步提升的潜力。

- **策略：**重点发展，通过定向推广和个性化推荐来激发其消费潜力。提供高额积分返还、特定消费类别的折扣等，鼓励其增加消费。

6. 重要价值用户 (17.52%)

- **特征：**这些用户是银行的核心用户，消费频率和金额都非常高。
- **策略：**维持并进一步提升其忠诚度。可以提供私人定制服务、高端理财产品、邀请参加高端客户活动等。建立一对一的 VIP 客服体系，确保这些用户得到最好的服务体验。

对于题目中的 id 为 211、3047、5971、7090、8576、9982 的客户，我们对其信用额度进行设置，结果如下：

表 11: 客户信用额度

id	**211**	**3047**	**5971**	**7090**	**8576**	**9982**
cred_limit	60000	80000	45000	20000	30000	65000

六　模型的评价与推广

6.1　模型的优点

1. 模型在交叉验证和测试中的表现都很优异，显示出较高的准确性和稳定性。
2. 过主成分分析（PCA）对数据进行降维，保留了主要特征，提高了模型的计算效率。
3. 使用 SMOTE 过采样技术有效解决了数据不平衡问题，提升了模型的预测能力。
4. 结合熵权法和 TOPSIS 方法，全面客观地评估客户价值，分类结果准确。
5. 通过随机森林集成多个决策树的预测结果，提高了模型的鲁棒性和可靠性。

6.2　模型的缺点

1. 尽管进行了降维处理，但整个建模过程仍然复杂，计算资源需求较高。
2. 模型高度依赖于数据的质量和完整性，数据缺失或不准确可能会显著影响预测效果。
3. 模型主要针对特定银行客户的数据，其适用性在其他领域或数据中需进一步验证。

6.3　模型的推广

模型的推广可以应用于不同行业和地域，通过调整参数和特征选择，使其适应不同客户行为和市场环境。此外，可以结合多源数据，如社交媒体、地理位置和客户反馈，提升预测能力。模型可以进一步与实时数据处理技术结合，构建实时更新的预测模型，实现更加精准的营销策略。通过引入自动化和智能化技术，开发自动更新和优化的系统，减少人工干预，提升效率。为了增强模型的解释性，可以结合可解释性机器学习技术，使业务人员更好地理解模型的预测结果和决策依据。持续引入最新的机器学习算法和技术，不断优化模型性能，定期进行模型评估和更新，确保在不断变化的市场环境中保持高效和准确。这些推广措施将使模型在更多行业和场景中发挥作用，为不同领域的客户价值评估和精准营销提供支持，提高业务效益。

参考文献

[1] Tarawneh A S , Hassanat A B A , Almohammadi K ,et al.SMOTEFUNA: Synthetic Minority Over-sampling Technique based on Furthest Neighbour Algorithm[J].IEEE Access, 2020, 8:1-15.

[2] 赵蔷. 主成分分析方法综述 [J]. 软件工程, 2016, 19(6):3.

[3] 方匡南, 吴见彬, 朱建平, 等. 随机森林方法研究综述 [J]. 统计与信息论坛, 2011, 26(3):7.

[4] 潘妮, 周术华. 基于熵权的改进的 TOPSIS 模型及其应用 [J]. 云南水力发电, 2007, 23(5):5.

[5] 伍维维. 基于组合赋权法的 RFMN 模型的信用卡用户价值分析 [D]. 重庆工商大学, 2021.

[6] 常彤.K-means 算法及其改进研究现状 [J]. 通讯世界, 2017(19):2.

附录 1 问题一代码

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 设置字体
plt.rcParams['font.sans-serif'] = ['SimHei']

# Adobe配色方案
adobe_colors = ['#1A4F63', '#068587', '#6FB07F', '#EFC050', '#FCB03C',
                '#FF7A48', '#FC5B3F', '#D65076', '#DD4124', '#D65076']

# 读取数据
file_path = 'handledData.xlsx'
df = pd.read_excel(file_path)


bins = [0,500,2000,5000,10000,30000,50000,float('inf')]
labels = ['0-500','500-2000', '2000-5000', '5000-10000', '10000-30000', '30000-50000', '50000以上']
df['consume_num_session3_group'] = pd.cut(df['consume_amt_session'], bins=bins, labels=labels, right=False)

# 按分组汇总 y 列的值
grouped_df = df.groupby(['consume_num_session3_group', 'y']).size().unstack(fill_value=0)

# 计算每个区间内分期用户和非分期用户的总数
total_instalment_users = grouped_df[1].sum()
total_non_instalment_users = grouped_df[0].sum()
print('每个区间内分期用户和非分期用户的总数')
print(total_instalment_users)
print(total_non_instalment_users)
# 计算分期用户和非分期用户的百分比
instalment_percentages = grouped_df[1] / total_instalment_users * 100
non_instalment_percentages = grouped_df[0] / total_non_instalment_users * 100
print('分期用户和非分期用户的百分比')
print(grouped_df[1])
print(grouped_df[0])
# 绘制饼图
fig, axes = plt.subplots(1, 2, figsize=(12, 7))

ax1 = axes[0]
instalment_percentages.plot(kind='pie', autopct='%1.1f%%', startangle=90, ax=ax1, colors=adobe_colors)
ax1.set_title(f'分期用户的本期账单累计消费金额\n总数: {total_instalment_users}')

ax2 = axes[1]
non_instalment_percentages.plot(kind='pie', autopct='%1.1f%%', startangle=90, ax=ax2, colors=adobe_colors)
```

```
ax2.set_title(f'非分期用户的本期账单累计消费金额\n总数: {total_non_instalment_users}')

plt.tight_layout()
plt.figure(dpi=300)
plt.show()

# 绘制箱线图
plt.figure(figsize=(10, 6))
sns.boxplot(x='y', y='six_cycle_mp_avg_amt', data=df, palette=['#E89275', '#9CBCB7'])
plt.title('额度的分期用户与未分期用户')
plt.xlabel('是否办理交易分期（1=是，0=否）')
plt.ylabel('年龄')
legend = plt.legend(labels=['未分期用户', '分期用户'], loc='upper right')
legend.legendHandles[0].set_color('#E89275')
legend.legendHandles[1].set_color('#9CBCB7')
plt.figure(dpi=300)
plt.show()

# 计算各区间的分期与未分期的占比
consume_num_session3_group_counts = grouped_df.div(grouped_df.sum(axis=1), axis=0) * 100

# 绘制条形图
plt.figure(figsize=(10, 8))

bar_width = 0.4
index = consume_num_session3_group_counts.index
bar1 = plt.barh(index, consume_num_session3_group_counts[0], height=bar_width, label='未分期用户', color='
    #0078AE')
bar2 = plt.barh(index, consume_num_session3_group_counts[1], height=bar_width, left=
    consume_num_session3_group_counts[0], label='分期用户', color='#F29985')

plt.xlabel('百分比')
plt.ylabel('本期账单累计消费金额')
plt.title('各本期账单累计消费金额分期与未分期用户百分比')
plt.legend()
plt.xlim(0, 100)
plt.tight_layout()
plt.figure(dpi=300)
plt.show()
```

附录 2 问题二代码

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE
from sklearn.metrics import accuracy_score, mean_squared_error, f1_score, recall_score, precision_score, roc_curve, auc
from sklearn.model_selection import train_test_split, cross_val_score
import matplotlib.pyplot as plt

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


def train_random_forest_with_pca(data, oversample=True, pca_components=15, random_forest_params=None, random_state=25):
    X = data.iloc[:, 3:]
    y = data['y']

    X['last_mp_days'].replace(9999, 999999, inplace=True)

    if oversample:
        smote = SMOTE(random_state=random_state)
        X_resampled, y_resampled = smote.fit_resample(X, y)
    else:
        X_resampled, y_resampled = X, y

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X_resampled)

    pca = PCA(n_components=pca_components, random_state=random_state)
    X_pca = pca.fit_transform(X_scaled)

    if random_forest_params is None:
        random_forest_params = {'n_estimators': 200, 'max_depth': 30, 'random_state': random_state}

    rf_classifier = RandomForestClassifier(**random_forest_params)
    rf_classifier.fit(X_pca, y_resampled)

    train_accuracy = accuracy_score(y_resampled, rf_classifier.predict(X_pca))

    return rf_classifier, pca, scaler, train_accuracy


def predict_customer_intent(customer_data, model, pca, scaler):
```

```
    X_customer = customer_data.iloc[:, 3:]

    X_customer['last_mp_days'].replace(9999, 999999, inplace=True)

    X_customer_scaled = scaler.transform(X_customer)
    X_customer_pca = pca.transform(X_customer_scaled)

    intent_probabilities = model.predict_proba(X_customer_pca)[:, 1]

    results = []
    for i, prob in enumerate(intent_probabilities):
        results.append({
            '客户ID': customer_data.iloc[i, 1],
            '原始意向': customer_data.iloc[i, 2],
            '预测概率': prob
        })

    return results


# 加载数据
customer_data = pd.read_excel('customer_data.xlsx')
handled_data = pd.read_excel('handledData.xlsx')

# 划分数据集为训练集和测试集
X = handled_data.iloc[:, 3:]
y = handled_data['y']
X['last_mp_days'].replace(9999, 999999, inplace=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=0, stratify=y)

# 进行SMOTE过采样
smote = SMOTE(random_state=0)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# 特征缩放
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_resampled)
X_test_scaled = scaler.transform(X_test)

# PCA降维
pca = PCA(n_components=20, random_state=0)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

# 随机森林模型训练
rf_classifier = RandomForestClassifier(n_estimators=200, max_depth=30, random_state=0)
```

```
rf_classifier.fit(X_train_pca, y_train_resampled)

# 使用交叉验证评估模型
cv_scores = cross_val_score(rf_classifier, X_train_pca, y_train_resampled, cv=10)
cv_mse_scores = cross_val_score(rf_classifier, X_train_pca, y_train_resampled, cv=10, scoring='
    neg_mean_squared_error')

# 输出交叉验证结果
print(f"交叉验证准确率: {cv_scores.mean()} ± {cv_scores.std()}")
print(f"交叉验证MSE: {-cv_mse_scores.mean()} ± {cv_mse_scores.std()}")

# 在测试集上评估模型
test_predictions = rf_classifier.predict(X_test_pca)
test_probabilities = rf_classifier.predict_proba(X_test_pca)[:, 1]

test_accuracy = accuracy_score(y_test, test_predictions)
test_mse = mean_squared_error(y_test, test_predictions)
test_f1 = f1_score(y_test, test_predictions)
test_recall = recall_score(y_test, test_predictions)
test_precision = precision_score(y_test, test_predictions)

print(f"测试集准确率: {test_accuracy}")
print(f"测试集MSE: {test_mse}")
print(f"测试集F1分数: {test_f1}")
print(f"测试集召回率: {test_recall}")
print(f"测试集精确率: {test_precision}")

# 预测客户意愿
customer_intent = predict_customer_intent(customer_data, rf_classifier, pca, scaler)

# 输出每个客户的预测结果
for result in customer_intent:
    print(f"客户ID: {result['客户ID']}, 原始意向: {result['原始意向']}, 预测概率: {result['预测概率']}")
```

附录 3 问题三代码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
plt.rcParams['font.sans-serif'] = ['SimHei']

# 读取数据
file_path = 'handledData.xlsx'
data = pd.read_excel(file_path, sheet_name='Sheet1')
# 去除cred_limit列中等于1的异常值
data = data[data['cred_limit'] != 1]

# 定义各个因素的数据
factor1_data = data[['six_bill_avg_amt', 'consume_amt_session12', 'consume_amt_session6', '
    consume_amt_session3', 'consume_amt_session', 'six_cycle_mp_avg_amt']]
factor2_data = data[['six_bill_low_repay_num']]
factor3_data = data[['this_bill_rate', 'month_avg_use_year', 'month_avg_use_month6', 'month_avg_use_month3']]
factor4_data = data[['consume_num_session12', 'consume_num_session6', 'consume_num_session3', '
    consume_num_session', 'six_bill_num']]
factor5_data = data[['six_bill_avg_debt_rate']]
factor6_data = data[['xaccount_age', 'six_cycle_mp_num']]

# 对负向指标取反
factor2_data = -factor2_data
factor5_data = -factor5_data

# 标准化所有因素的数据
scaler = MinMaxScaler()
factor1_data_scaled = scaler.fit_transform(factor1_data)
factor2_data_scaled = scaler.fit_transform(factor2_data)
factor3_data_scaled = scaler.fit_transform(factor3_data)
factor4_data_scaled = scaler.fit_transform(factor4_data)
factor5_data_scaled = scaler.fit_transform(factor5_data)
factor6_data_scaled = scaler.fit_transform(factor6_data)

# PCA降维函数
def apply_pca(data):
    pca = PCA(n_components=1)
    transformed_data = pca.fit_transform(data)
    explained_variance = pca.explained_variance_ratio_[0]
    components = pca.components_[0]
    return transformed_data, explained_variance, components

```

```
# 对每个因素进行PCA降维
factor1_data_pca, factor1_explained_variance, factor1_components = apply_pca(factor1_data_scaled)
factor2_data_pca, factor2_explained_variance, factor2_components = apply_pca(factor2_data_scaled)
factor3_data_pca, factor3_explained_variance, factor3_components = apply_pca(factor3_data_scaled)
factor4_data_pca, factor4_explained_variance, factor4_components = apply_pca(factor4_data_scaled)
factor5_data_pca, factor5_explained_variance, factor5_components = apply_pca(factor5_data_scaled)
factor6_data_pca, factor6_explained_variance, factor6_components = apply_pca(factor6_data_scaled)

# 合并所有因素的PCA降维数据
all_factors_data_pca = np.hstack([factor1_data_pca, factor2_data_pca, factor3_data_pca, factor4_data_pca,
    factor5_data_pca, factor6_data_pca])

# 再次标准化PCA降维后的数据
standardized_data = scaler.fit_transform(all_factors_data_pca)

m, n = standardized_data.shape
p = standardized_data / standardized_data.sum(axis=0)
entropy = -np.sum(p * np.log(p + 1e-10), axis=0) / np.log(m)

d = 1 - entropy
weights = d / d.sum()

factor_columns = ['factor1', 'factor2', 'factor3', 'factor4', 'factor5', 'factor6']
weight_dict = {factor_columns[i]: weights[i] for i in range(len(factor_columns))}
print("各个指标的权重:")
for factor, weight in weight_dict.items():
    print(f"{factor}: {weight:.4f}")

weighted_data = standardized_data * weights

ideal_solution = weighted_data.max(axis=0)
negative_ideal_solution = weighted_data.min(axis=0)

distance_to_ideal = np.sqrt(((weighted_data - ideal_solution) ** 2).sum(axis=1))
distance_to_negative_ideal = np.sqrt(((weighted_data - negative_ideal_solution) ** 2).sum(axis=1))

scores = distance_to_negative_ideal / (distance_to_ideal + distance_to_negative_ideal)
data['score'] = scores

data['score'].fillna(0, inplace=True)

data['normalized_score'] = scaler.fit_transform(data[['score']])

plt.hist(data['normalized_score'], bins=10, edgecolor='black')
plt.xlabel('客户综合价值得分')
plt.ylabel('客户数量')
```

```
plt.title('客户综合价值评分直方图')
plt.show()

data['user_category'] = pd.cut(data['normalized_score'], bins=[0, 0.5, 0.7, 1], labels=['普通用户', '白银用户'
    , '黄金用户'])

user_counts = data['user_category'].value_counts()
user_ratios = data['user_category'].value_counts(normalize=True)

print("各类用户数量:")
print(user_counts)
print("\n各类用户占比:")
print(user_ratios)

plt.figure(figsize=(8, 6))
plt.pie(user_counts, labels=user_counts.index, autopct='%1.1f%%', startangle=140, colors=['lightblue', 'gold'
    , 'silver'])
plt.title('各类用户分布图')
plt.axis('equal')
plt.show()
```

附录 4 问题四代码

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

def getFeatures(factor):
    scaler = StandardScaler(with_mean=False, with_std=True)
    df_scaled = scaler.fit_transform(factor)
    pca = PCA(n_components=1)
    transformed_data = pca.fit_transform(df_scaled)
    print(pca.explained_variance_ratio_)
    return transformed_data

def calculate_entropy(series):
    n = len(series)
    probabilities = series / series.sum(axis=0)
    entropy = -sum(probabilities * np.log(probabilities + 1e-10)) / np.log(n) # 避免对数为负无穷
    return entropy

df = pd.read_excel('./handledData.xlsx')
df['last_mp_days'] = 9999 - df['last_mp_days']
factor_R = df[["last_mp_days",'xaccount_age']]
factor_F = df[['consume_num_session12', 'consume_num_session6', 'consume_num_session3',
               'consume_num_session', 'six_bill_num', 'six_cycle_mp_num', 'epp_nbr_12m'
               ]]
factor_M = df[['six_bill_avg_amt', 'consume_amt_session12', 'consume_amt_session6',
               'consume_amt_session3', 'consume_amt_session', 'six_cycle_mp_avg_amt'
               ]]
factor_N = df[['six_bill_low_repay_num', 'six_bill_avg_debt_rate']]

feature_R = getFeatures(factor_R)
feature_F = getFeatures(factor_F)
feature_M = getFeatures(factor_M)
feature_N = getFeatures(factor_N)

feature1 = pd.DataFrame(feature_R, columns=['feature'])
feature2 = pd.DataFrame(feature_F, columns=['feature'])
feature3 = pd.DataFrame(feature_M, columns=['feature'])
feature4 = pd.DataFrame(feature_N, columns=['feature'])

quantiles = feature1['feature'].quantile([0, 0.2, 0.4, 0.6, 0.8, 1])
bins = [-np.inf, quantiles[0.2], quantiles[0.4], quantiles[0.6], quantiles[0.8], np.inf]
```

```
feature1['quintile'] = np.digitize(feature1['feature'], bins)
quantiles = feature2['feature'].quantile([0, 0.2, 0.4, 0.6, 0.8, 1])
bins = [-np.inf, quantiles[0.2], quantiles[0.4], quantiles[0.6], quantiles[0.8], np.inf]
feature2['quintile'] = np.digitize(feature2['feature'], bins)
quantiles = feature3['feature'].quantile([0, 0.2, 0.4, 0.6, 0.8, 1])
bins = [-np.inf, quantiles[0.2], quantiles[0.4], quantiles[0.6], quantiles[0.8], np.inf]
feature3['quintile'] = np.digitize(feature3['feature'], bins)
quantiles = feature4['feature'].quantile([0, 0.2, 0.4, 0.6, 0.8, 1])
bins = [-np.inf, quantiles[0.2], quantiles[0.4], quantiles[0.6], quantiles[0.8], np.inf]
feature4['quintile'] = np.digitize(feature4['feature'], bins)
feature4['quintile'] = feature4['quintile'].map({1: 5, 2: 4,3:3, 4: 2, 5: 1})

scaler = MinMaxScaler()
feature1['feature'] = scaler.fit_transform(feature1['feature'].to_numpy().reshape(-1, 1))
feature2['feature'] = scaler.fit_transform(feature2['feature'].to_numpy().reshape(-1, 1))
feature3['feature'] = scaler.fit_transform(feature3['feature'].to_numpy().reshape(-1, 1))
feature4['feature'] = scaler.fit_transform(feature4['feature'].to_numpy().reshape(-1, 1))

entropies_R = calculate_entropy(feature1['feature'])
entropies_F = calculate_entropy(feature2['feature'])
entropies_M = calculate_entropy(feature3['feature'])
entropies_N = calculate_entropy(feature4['feature'])

difference_coefficients = 1 - entropies_R + 1 - entropies_F + 1 - entropies_M + 1 - entropies_N
weights_R = (1 - entropies_R) / difference_coefficients
weights_F = (1 - entropies_F) / difference_coefficients
weights_M = (1 - entropies_M) / difference_coefficients
weights_N = (1 - entropies_N) / difference_coefficients

part1_R = feature1['feature'].sum() * weights_R
part1_F = feature2['feature'].sum() * weights_F
part1_M = feature3['feature'].sum() * weights_M
part1_N = feature4['feature'].sum() * weights_N
part1 = part1_R + part1_F + part1_M + part1_N

weights2_R, weights2_F, weights2_M, weights2_N = 0.099, 0.345, 0.370, 0.185
part2_R = feature1['feature'].sum() * weights2_R
part2_F = feature2['feature'].sum() * weights2_F
part2_M = feature3['feature'].sum() * weights2_M
part2_N = feature4['feature'].sum() * weights2_N
part2 = part2_R + part2_F + part2_M + part2_N

T = part2 / (part1 + part2)
U = part1 / (part1 + part2)
weights3_R = weights2_R * T + weights_R * U
weights3_F = weights2_F * T + weights_R * U
```

```
weights3_M = weights2_M * T + weights_M * U
weights3_N = weights2_N * T + weights_N * U

feature1['R_S'] = feature1['quintile'] * weights3_R
feature2['F_S'] = feature2['quintile'] * weights3_F
feature3['M_S'] = feature3['quintile'] * weights3_M
feature4['N_S'] = feature4['quintile'] * weights3_N
feature = pd.concat([feature1['R_S'], feature2['F_S'], feature3['M_S'], feature4['N_S']], axis=1)
feature['S'] = feature['R_S'] + feature['F_S'] + feature['M_S'] + feature['N_S']

k_values = range(2,9)
sse_scores = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(feature)
    sse_scores.append(kmeans.inertia_)

# 绘制Elbow图
plt.plot(k_values, sse_scores, marker='o', label='SSE')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.title('Elbow Method For Optimal k')
plt.legend()
plt.show()

kmeans = KMeans(n_clusters=6, random_state=42)
kmeans.fit(feature)
cluster_centers = kmeans.cluster_centers_
labels = kmeans.labels_

clustered_data = {}
for i in range(6): # 假设有6个簇
    clustered_data[i] = feature[kmeans.labels_ == i]
for i, cluster in clustered_data.items():
    # print(f"\nCluster {i}:")
    # print(cluster.index)
    # print(df.loc[cluster.index, :].describe()) # 211 3047 5971 7090 8576 9982
    if 36 in cluster.index:
        print(i)
        print(df.loc[36, ['id', 'cred_limit']])
        print(df.loc[36, 'cred_limit'])
    if 500 in cluster.index:

        print(i)
        print(df.loc[500, ['id', 'cred_limit']])
```

```
    if 948 in cluster.index:
        print(i)
        print(df.loc[948, ['id', 'cred_limit']])
    if 8986 in cluster.index:
        print(i)
        print(df.loc[8986, ['id', 'cred_limit']])
    if 9319 in cluster.index:
        print(i)
        print(df.loc[9319, ['id', 'cred_limit']])
    if 9580 in cluster.index:
        print(i)
        print(df.loc[9580, ['id', 'cred_limit']])
```