
# Python语言

## 标题：基于机器学习的金融产品认购预测

姓名：王振宽

学号：22020040149

专业：应用统计

学院：统计与数学学院

**2022 年 12 月 17 日**

# 目录

# 1　引言

近些年来，随着互联网和大数据的蓬勃发展，金融机构越来越重视对客户数据进行信息的提取与挖掘。通过查阅相关资料可以得知，随着营销活动的增长，活动的影响力对公众而言已大大地削弱，在竞争压力与经济发展现状下，营销模式从活动转向了直销，即面向大众直接销售。通常情况下，需要对同一客户进行一次以上的联系，以便了解该金融产品是否会被认购，因此需要利用数据挖掘算法建立客户模型，数据集分为三个部分，第一类是客户的基本情况，包括年龄、学历、婚姻状态等等；第二部分是银行与客户的接触情况，包括联系次数、上次联系时间、联系方式等等；第三部分是社会经济环境，包括就业变化率、消费者价格指数等等。

## 1.1　问题的提出

根据给定的数据集，找到改进下一次营销活动的最佳策略，分析什么指标对客户认购产品的影响较大，金融机构如何才能在未来的营销活动中发挥更大的作用？

为了回答这个问题，我们必须分析金融机构的营销活动，并确定有助于我们得出结论以指定未来战略的模式，并对新数据集进行预测，判断那个是潜在的客户。

# 2　模型的假设

- 数据集的标签数据真实可靠；
- 数据集各个特征之间是相互独立的；
- 类别中的“unknown”不是缺失值和异常值；
- 数据集中的类别比较全面，不存在其他未知的情况。

# 3　数据分析

## 3.1　数据集说明

数据是与某个金融机构的营销活动有关的数据集，共计有26645条记录，每条记录包括19条客户基本信息变量，1个是否认购该金融产品，作为我们的目标变量。影响客户是否认购该金融产品的因素主要有19个，我们可以将其分为3类，包括客户基本情况、金融机构与客户接触情况、社会经济环境，具体变量描述如下：

**表1：数据集描述**

| 类别 | 变量名称 | 变量解释 |
|---|---|---|
| | age | 年龄(数值型) |
| | job | 工作情况 |
| | marital | 婚姻状况 |

| | | |
|---|---|---|
| 客户数据 | education | 教育水平 |
| | default | 信用是否已经违约 |
| | housing | 是否有住房贷款 |
| | loan | 是否有个人贷款 |
| 金融机构与客户接触情况 | contact | 联系时通信类型 |
| | month | 最后一次联系的月份 |
| | dayofweek | 最后一次联系是周几 |
| | campaign | 与该客户的联系次数 |
| | passed_days | 上次联系客户后的天数 |
| | previous | 与该客户的联系次数 |
| | pre_outcome | 前期营销活动的结果 |
| 社会经济环境 | emp_rate | 就业变化率 |
| | cpi | 消费者价格指数 |
| | cci | 消费者信心指数 |
| | r3m | 银行的 3 个月利率 |
| | employed | 雇员人数 |

## 3.2 数据预处理

拿到我们的数据后，我们先对数据分布有所了解，需要进行描述性统计分析。主要操作是检测数据是否存在缺失值或异常值，若存在以上问题，则进行相关处理，希望达到提高数据质量的目的，有助于后续建立更优良的模型。

我们的数据集维度为 $26645\times20$，正标签的数量为 23596，负标签的数量为 3049。数据分布还是很不平衡的，后边可以使用过采样的方法解决样本不均衡问题。

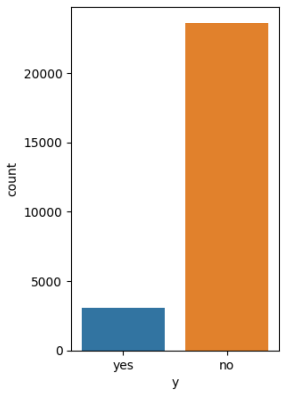


**图1：数据集分布情况**

检查一下数据集中是否存在空值，结果如下：

表2：数据集是否存在空值

| Column | Non-Null Count | Dtype |
|---|---|---|
| age | 26645 non-null | int64 |
| job | 26645 non-null | object |
| marital | 26645 non-null | object |
| education | 26645 non-null | object |
| default | 26645 non-null | object |
| housing | 26645 non-null | object |
| loan | 26645 non-null | object |
| contact | 26645 non-null | object |
| month | 26645 non-null | object |
| day_of_week | 26645 non-null | object |
| campaign | 26645 non-null | int64 |
| passed_days | 26645 non-null | int64 |
| previous | 26645 non-null | int64 |
| pre_outcome | 26645 non-null | object |
| emp_rate | 26645 non-null | float64 |
| cpi | 26645 non-null | float64 |
| cci | 26645 non-null | float64 |
| r3m | 26645 non-null | float64 |
| employed | 26645 non-null | float64 |
| y | 26645 non-null | object |

我们对数据集中的各类变量进行描述性分析，首先是数值型变量：

表3：数值型变量的描述性分析

| | age | campaign | passed_days | previous | emp_rate | cpi | cci | r3m | employed |
|---|---|---|---|---|---|---|---|---|---|
| count | 26645 | 26645 | 26645 | 26645 | 26645 | 26645 | 26645 | 26645 | 26645 |
| mean | 39.98 | 2.56 | 962.85 | 0.17 | 0.08 | 93.57 | -40.48 | 3.62 | 5167.17 |
| std | 10.41 | 2.77 | 186.00 | 0.49 | 1.57 | 0.58 | 4.63 | 1.73 | 72.21 |
| min | 17.00 | 1.00 | 0 | 0 | -3.40 | 92.20 | -50.80 | 0.63 | 4963.60 |
| 25% | 32.00 | 1.00 | 999 | 0 | -1.80 | 93.07 | -42.70 | 1.34 | 5099.10 |
| 50% | 38.00 | 2.00 | 999 | 0 | 1.10 | 93.75 | -41.80 | 4.85 | 5191.00 |
| 75% | 47.00 | 3.00 | 999 | 0 | 1.40 | 93.99 | -36.40 | 4.96 | 5228.10 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| max | 98.00 | 43.00 | 999 | 7 | 1.40 | 94.77 | -26.90 | 5.04 | 5228.10 |

表4：分布型变量的描述性分析

| job | 数量 | marital | 数量 | education | 数量 |
|---|---|---|---|---|---|
| admin. | 6703 | married | 16152 | university.degree | 7849 |
| blue-collar | 6021 | single | 7485 | high.school | 6238 |
| technician | 4376 | divorced | 2959 | basic.9y | 3890 |
| services | 2530 | unknown | 49 | professional.course | 3355 |
| management | 1940 | | | basic.4y | 2665 |
| retired | 1115 | | | basic.6y | 1508 |
| self-employed | 937 | | | unknown | 1127 |
| entrepreneur | 922 | | | illiterate | 13 |
| housemaid | 671 | | | | |
| unemployed | 639 | | | | |
| student | 565 | | | | |
| unknown | 226 | | | | |
| default | 数量 | housing | 数量 | loan | 数量 |
| no | 21096 | yes | 13971 | no | 21943 |
| unknown | 5547 | no | 12018 | yes | 4046 |
| yes | 2 | unknown | 656 | unknown | 656 |
| month | 数量 | day_of_week | 数量 | pre_outcome | 数量 |
| may | 8876 | thu | 5644 | nonexistent | 23072 |
| jul | 4712 | mon | 5455 | failure | 2695 |
| aug | 3999 | wed | 5322 | success | 878 |
| jun | 3409 | tue | 5223 | | |
| nov | 2654 | fri | 5001 | | |
| apr | 1686 | | | | |
| oct | 472 | | | | |
| sep | 375 | | | | |
| mar | 346 | | | | |
| dec | 116 | | | | |

从上面的表格中我们不难看出，原始数据都没有明显的缺失值。

## 3.3 数据可视化

我们来看以下不同特征对客户认购情况的影响。

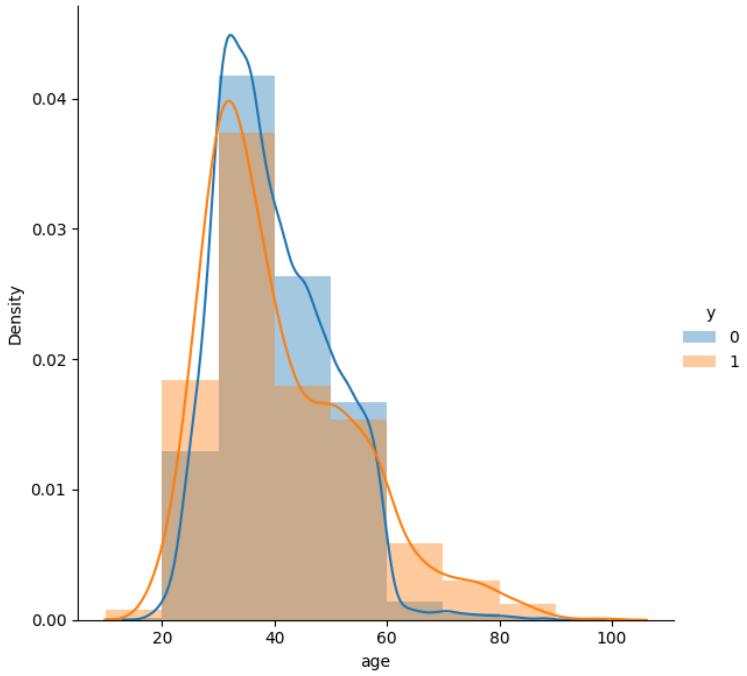


**图2：认购和未认购的年龄分布**

由age分布图可知，数据主要分布在20岁到60岁，并且数据存在右偏，即20岁以下和60岁以上年龄层次的客户认购的较少。

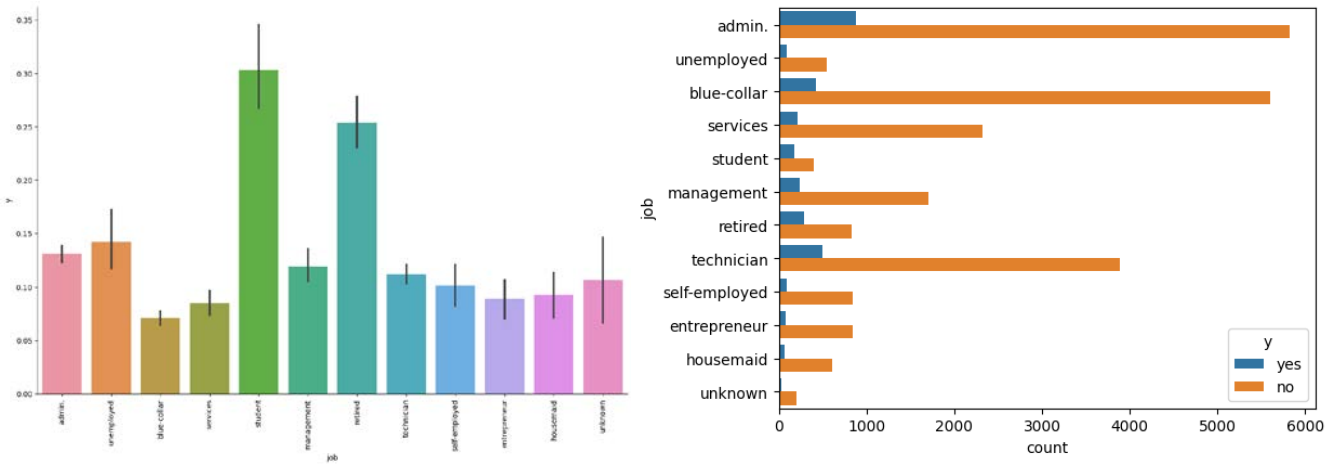


**图3：job分布图**

退休人员和学生最青睐认购该产品，其次是失业人员，蓝领认购产品的比率最低。

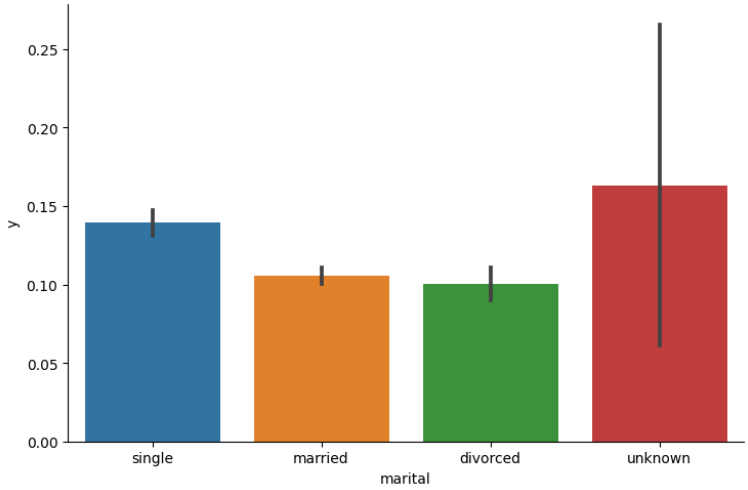


**图4：婚姻状况与认购的关系**

除去未知的婚姻状态，单身相比离异和已婚更倾向于认购该产品。

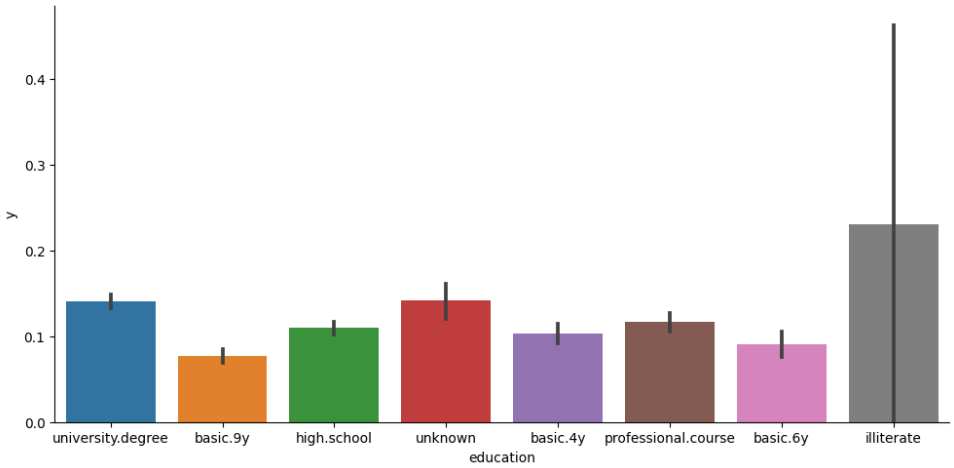


**图5：学历和认购产品的关系**

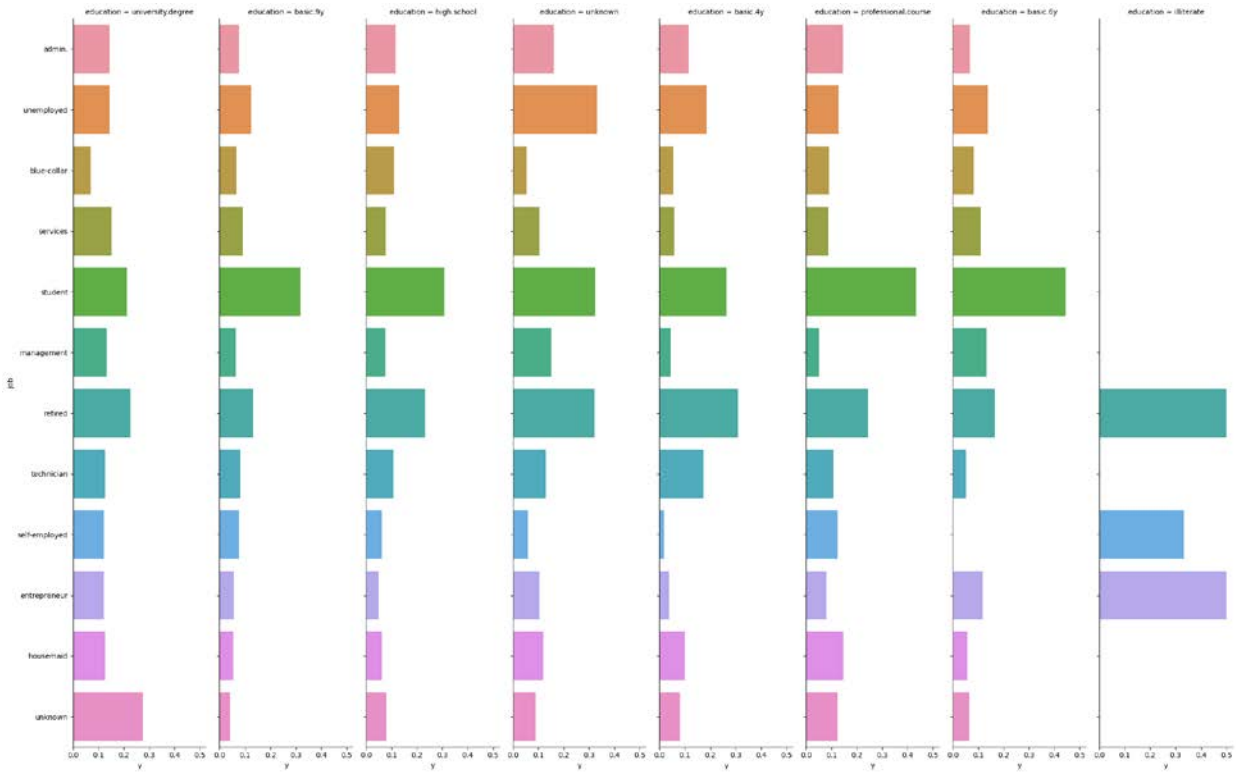


**图6：不同学历与认购产品的关系**

无论接受教育的程度如何，会认购该金融机构产品的人主要是退休人员和学生，其次是失业人员。随着教育程度的提高，管理人员、技术人员、行政人员、蓝领、企业家、自由职业者、事业人员越来越倾向认购该金融机构的产品。

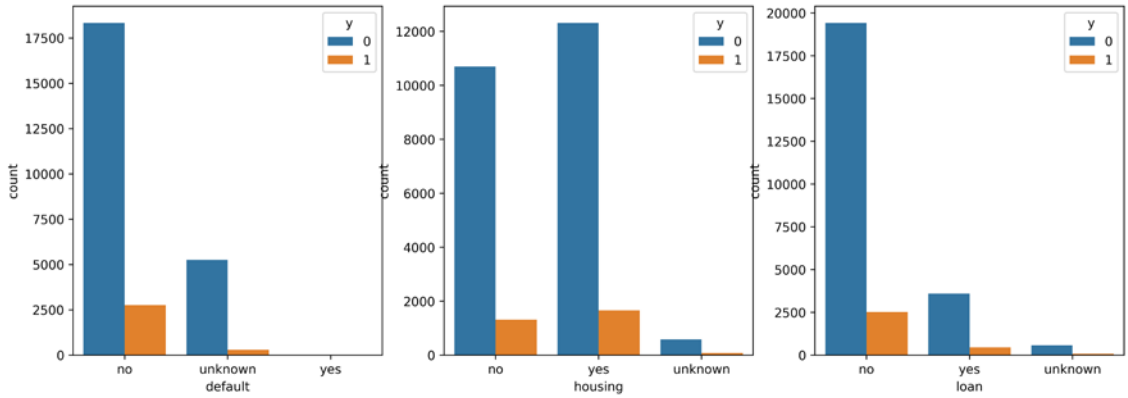


**图7：违约、住房、贷款与认购产品的数量关系**

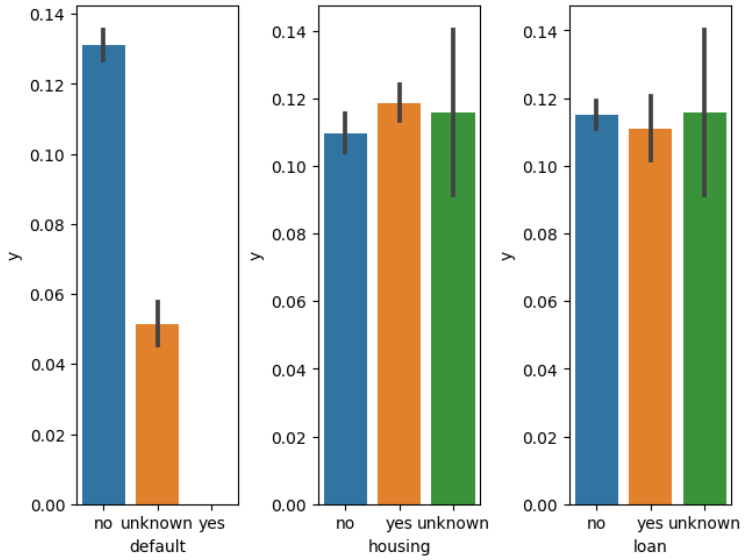


**图8：违约、住房、贷款与认购产品的关系**

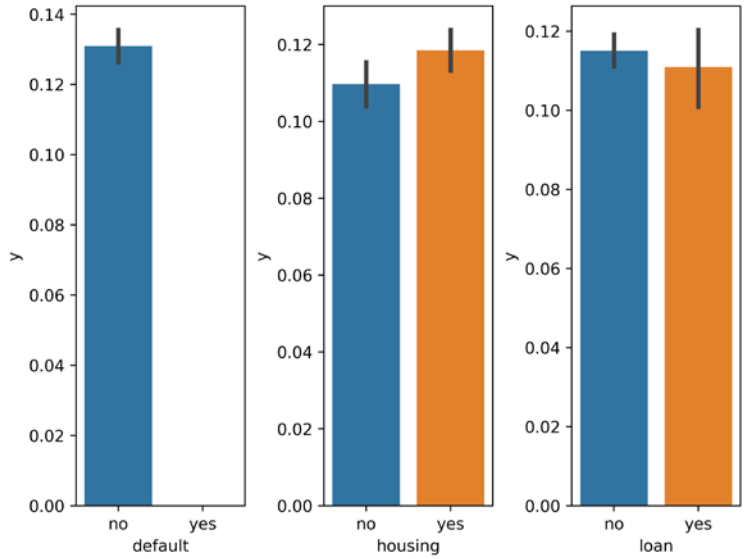


**图9：去掉“unknown”类型后三者与认购产品的关系**

没有违约的人倾向于认购该产品，没有个人贷款的人倾向于认购该产品，而是否有住房的

影响并不大。

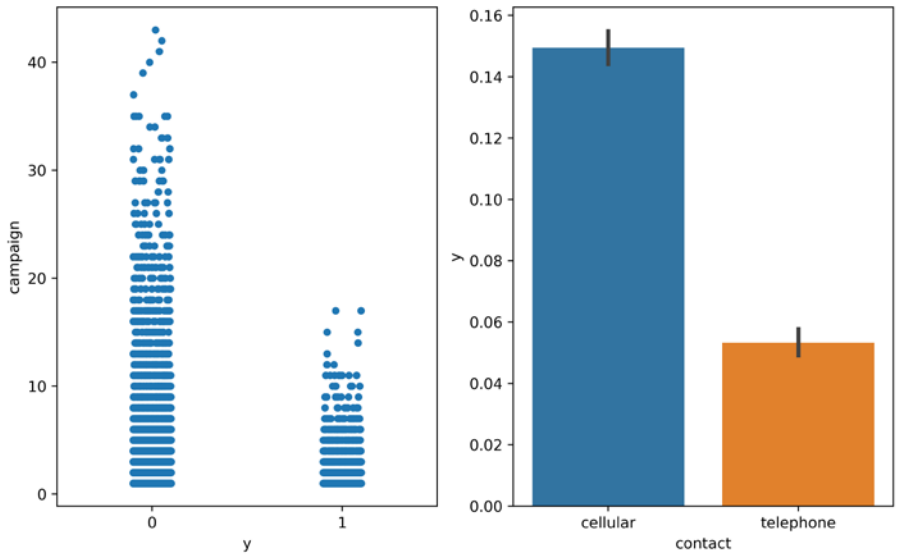


**图10：沟通方式和认购产品的关系**

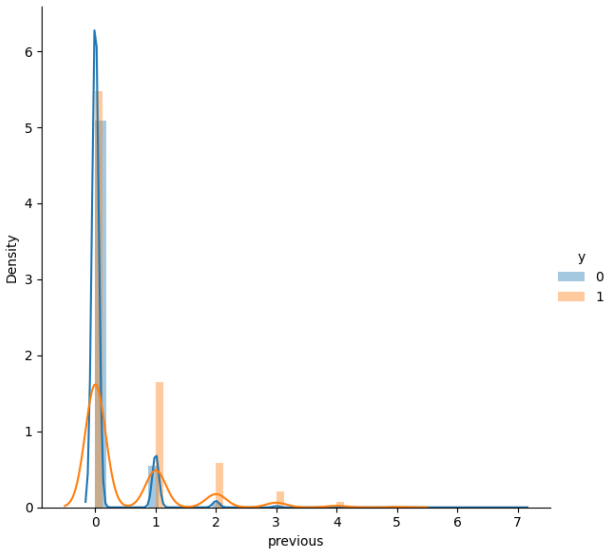


**图11：联系次数与认购产品的关系**

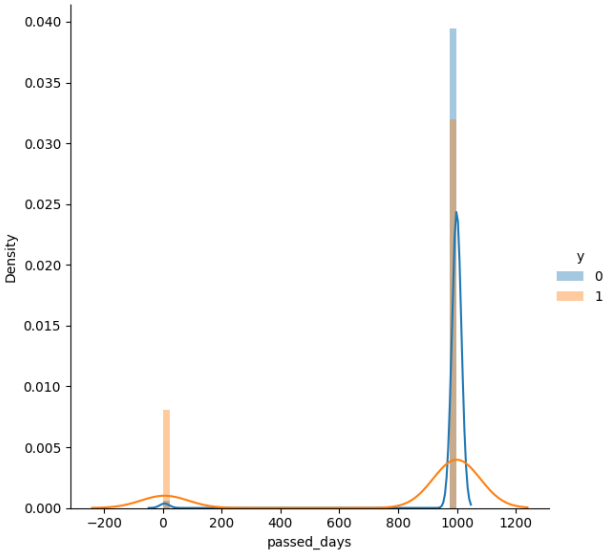


**图12：上次联系客户后的天数与认购的关系**

通过移动电话沟通客户认购产品的比率高，沟通次数长并不一定能够提高客户认购产品的比率，最后一次沟通的超过一定时长认购产品的比率提高。

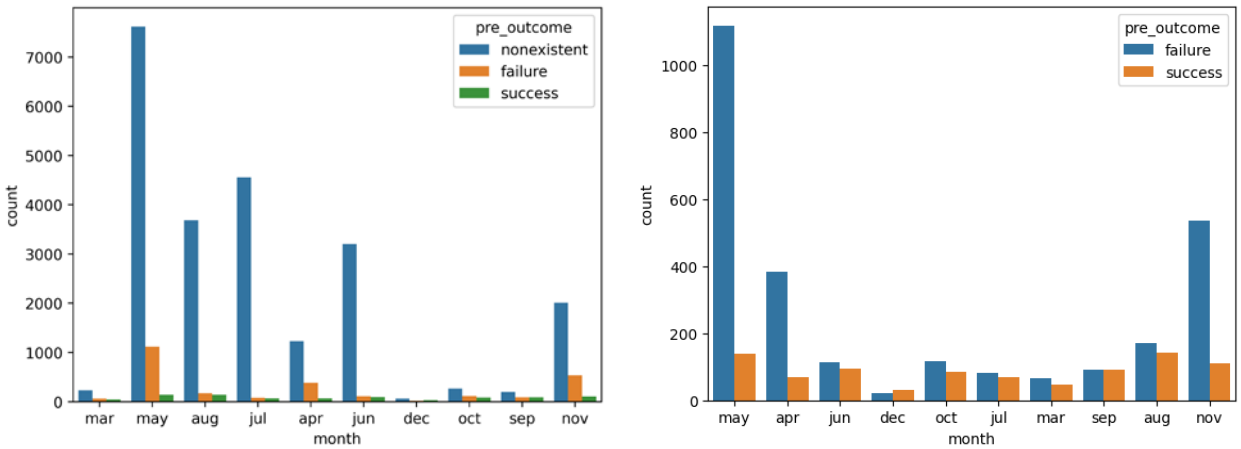


**图11：过去月份对认购产品的关系**

四月、五月和十一月的营销活动中，人们更倾向于拒绝认购该产品，下次营销活动中应该选择避开这三个月。

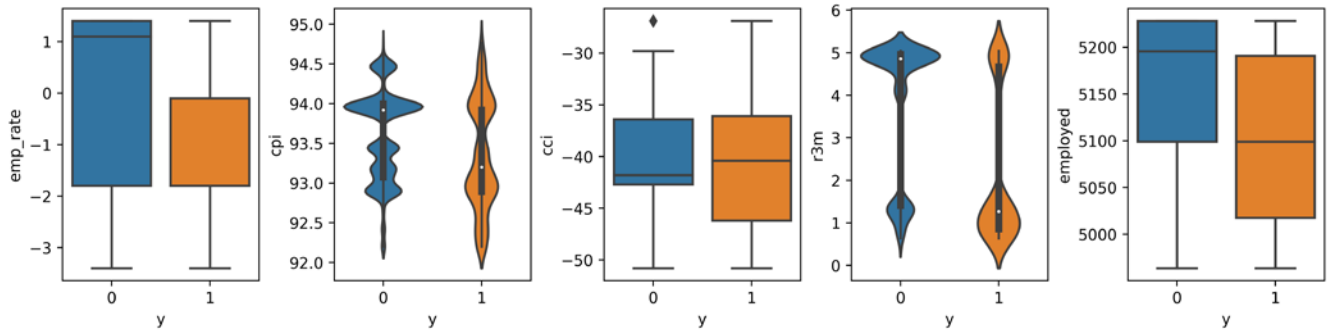


**图12：社会经济环境对认购产品的影响**

就业变化率低，消费者价格指数低，银行利率低，雇员人数减少的时候，人们更倾向于认购该产品。

# 4 数据建模

## 4.1 特征处理

对于有限的分类变量做onehot编码处理，我们看一下结果：

| | age | campaign | passed_days | previous | emp_rate | cpi | cci | r3m | employed | job_admin. | ... | month_sep | day_of_week_fri | day_of_week_mon |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 27 | 1 | 999 | 0 | -1.8 | 93.369 | -34.8 | 0.637 | 5008.7 | 1 | ... | 0 | 0 | 0 |
| 1 | 55 | 1 | 999 | 0 | -1.8 | 92.893 | -46.2 | 1.264 | 5099.1 | 0 | ... | 0 | 0 | 1 |
| 2 | 25 | 3 | 999 | 1 | -1.8 | 92.893 | -46.2 | 1.250 | 5099.1 | 0 | ... | 0 | 1 | 0 |
| 3 | 43 | 1 | 999 | 0 | 1.4 | 93.444 | -36.1 | 4.968 | 5228.1 | 1 | ... | 0 | 0 | 0 |
| 4 | 33 | 1 | 999 | 0 | -2.9 | 92.469 | -33.6 | 1.044 | 5076.2 | 1 | ... | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 26640 | 41 | 4 | 999 | 0 | 1.4 | 94.465 | -41.8 | 4.866 | 5228.1 | 1 | ... | 0 | 0 | 0 |
| 26641 | 35 | 1 | 999 | 1 | -0.1 | 93.200 | -42.0 | 4.076 | 5195.8 | 0 | ... | 0 | 0 | 0 |
| 26642 | 53 | 7 | 999 | 0 | 1.4 | 93.444 | -36.1 | 4.962 | 5228.1 | 0 | ... | 0 | 0 | 0 |
| 26643 | 35 | 2 | 999 | 0 | -1.8 | 93.075 | -47.1 | 1.445 | 5099.1 | 0 | ... | 0 | 0 | 0 |
| 26644 | 48 | 2 | 999 | 0 | -1.8 | 92.893 | -46.2 | 1.344 | 5099.1 | 0 | ... | 0 | 0 | 0 |

26645 rows × 63 columns

**图13：one-hot编码后的结果**

## 4.2 过采样

### 4.2.1 不均衡数据处理方法

由于数据的不平衡性，将导致小样本数据的信息量较少，而使预测结果往往偏向于数据集中的大样本数据，从而使得模型的对于小样本的预测精度大大降低，在整个模型的创建过程中，是以其整体误差率最小化为原则的，因此小样本在整个样本中所占比重很小，虽然小样本预测精度很差，但整体精度却表现良好，而我们更关注的是小样本的预测效果如何，因此，对于不平衡的数据仅仅观察整体误差率是不够的。

大部分的算法都假设数据是均衡的,算法中往往认为各类别的预测误差对于整体精度的损失是一样的，对于机器学习而言，对于不均衡数据的预测也不是很稳定，甚至有的结果很不理想，因此，机器学习也多基于均衡数据进行建模。

我们上面已经看到，数据集的分布不均匀：

**表5：数据集比例**

| y | counts | 比例 |
|---|---|---|
| yes | 3049 | 11.44% |
| no | 23596 | 88.56% |
| 总和 | 26645 | 100% |

不平衡数据的分类是指在响应变量中某一类的样本量远远大于其他类别，相较于多分类的响应指标，不平衡数据较多的出现于二分类中，在现实生活中，我们会发现在大学的招收里面，最终被录取的比例极低；在某些欺诈活动中，违规的交易数远大于合法交易；在健康普查中，患有重大疾病的远少于健康人数；在生产活动中，未达标的产品数远小于达标产品数，因此，对于这些数据极为不平衡的数据，当对如此常见不平衡数据进行分析时，便需要将其转化为均衡数据。

对于不平衡数据，我们可以利用采样法解决不平衡问题，利用欠采样、过采样等方法进行修正，改变数据集的分布情况，使得变量的分布更加的均衡。

（1） 欠采样法

欠采样法主要适用于数据量较大的情况下，对数据集中类别较大的样本进行采样，进而减少大类的数据量，以此来平衡整个样本类别的平衡性，同时也减少了对大样本数据集计算的成本花费。

但是需要注意的是，采用欠采样法是将观测值进行了删减，使得数据在一定程度上会有损失，特别是数据中的重要信息。

（2） 过采样

过采样法主要对数据集中类别较少的样本进行采样，经过多次重复抽样增加小类样本的数量，从而平衡各类别的数目。

过采样的优势是没有任何信息损失，但是由于对小类进行了重复抽样，因此造成小类中存在大量的重复数据，这可能会造成过拟合的情况，虽然在训练样本集中建模表现很好，但在预测未知数据时结果却不理想，同时过拟合增加了样本数据量，因此导致计算的成本也大大提高。

### 4.2.2 过采样后的数据结果

我们对数据集进行过采样，以期使得数据分布更加的均衡，此时的分布为：

**表6：过采样后的数据集分布情况**

| y | counts | 比例 |
|---|---|---|
| yes | 23596 | 50% |

| | | |
|---|---|---|
| no | 23596 | 50% |
| 总和 | 47192 | 50% |

## 4.3 数据集的划分和标准化

本文的主要目的是根据客户的属性特征建立模型，找到分类规则，进而可以对未知客户进行预测。

我们选取70%的数据作为训练集，剩下30%的数据作为测试集，并打乱数据集。

**表7：训练集和测试机的维度**

| 训练集 | 测试集 |
|---|---|
| (33034, 62) | (14158, 62) |

训练集的分布情况如下表所示：

**表8：训练集中样本分布情况**

| 类别 | counts | 比例 |
|---|---|---|
| 0(no) | 16551 | 50.1% |
| 1(yes) | 16483 | 49.9% |

测试集的分布情况如下表所示：

**表9：测试集中样本分布情况**

| 类别 | counts | 比例 |
|---|---|---|
| 0(no) | 7113 | 50.2% |
| 1(yes) | 7045 | 49.8% |

数据集划分好后我们对age、campaign、passed_days、previous、emp_rate、cpi、cci、r3m以及employed等特征进行标准化。

## 4.4 模型的建立

### 4.4.1 Adaboost模型

AdaBoost算法是一种有效而实用的Boosting算法，它以一种高度自适应的方法顺序地训练弱学习器。AdaBoost根据前一次的分类效果调整数据的权重，上一个弱学习器中错误分类样本的权重会在下一个弱学习器中增加，正确分类样本的权重会相应减少，并且在每一轮迭代时会向模型加入一个新的弱学习器。不断重复调整权重和训练弱学习器的过程，直到误分类数低于预设值或迭代次数达到指定最大迭代次数时，我们会得到一个强分类器。

算法的流程如下所示：

1. 先通过对 $N$ 个训练样本的学习得到第一个弱分类器；

2. 将分错的样本和其他的新数据一起构成一个新的 $N$ 个的训练样本，通过对这个样本的学习得到第二个弱分类器；

3. 将1和2都分错了的样本加上其他的新样本构成另一个新的 $N$ 个的训练样本，通过对这个样本的学习得到第三个弱分类器；

4. 最终经过提升的强分类器。即某个数据被分为哪一类要由各分类器权值决定。

由描述过程可知，该算法在实现过程中根据训练集的大小初始化样本权值，使其满足均匀分布，在后续操作中通过公式来改变和规范化算法迭代后样本的权值。样本被错误分类导致权值增大，反之权值相应减小，这表示被错分的训练样本集包括一个更高的权重。这就会使在下轮时训练样本集更注重于难以识别的样本，针对被错分样本的进一步学习来得到下一个弱分类器，直到样本被正确分类。在达到规定的迭代次数或者预期的误差率时，则强分类器构建完成。

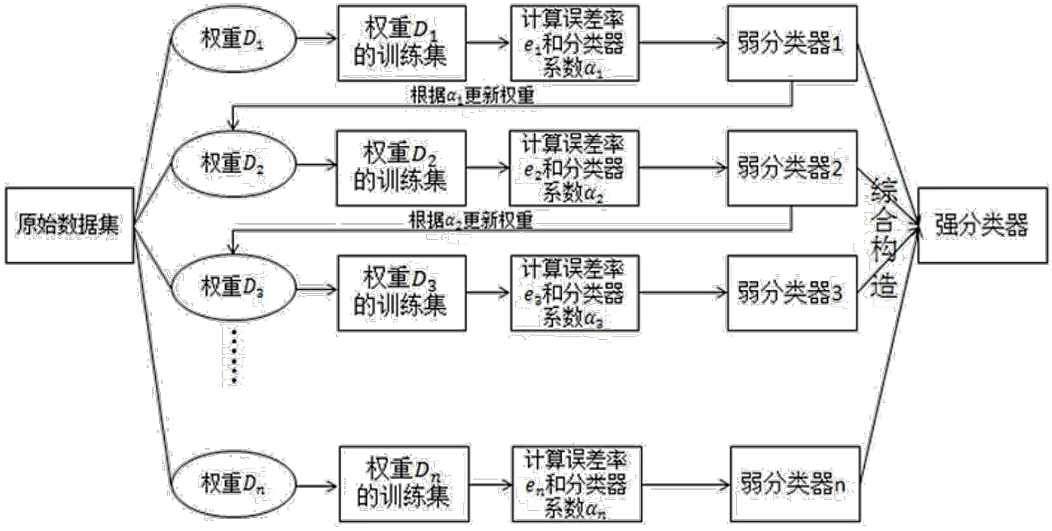


**图14：Adaboost算法流程**

Adaboost算法可以应用于模式识别、计算机视觉领域，用于二分类和多分类场景，其主要优点有：

1. 很好的利用了弱分类器进行级联；

2. 可以将不同的分类算法作为弱分类器；

3. AdaBoost具有很高的精度；

4. 相对于bagging算法和Random Forest算法，AdaBoost充分考虑的每个分类器的权重。

Adaboost算法的主要缺点有：

1. AdaBoost迭代次数也就是弱分类器数目不太好设定，可以使用交叉验证来进行确定；

2. 数据不平衡导致分类精度下降；

3. 训练比较耗时，每次重新选择当前分类器最好切分点。

### 4.3.2 Logistic 回归

逻辑回归（Logistic Regression）主要解决二分类问题，用来表示某件事情发生的可能性。

比如：一封邮件是垃圾邮件的可能性。

逻辑回归属于判别式模型，同时伴有很多模型正则化的方法，而且不需要担心特征是否相关。与决策树、支持向量机相比，能得到一个不错的概率表示。它使用的激活函数为Sigmoid函数，表达式如下：

$$f(x)=\frac{1}{1+e^{-x}}$$

对于一个二分类问题，设因变量为 $P(y=1|x)=p$ ，对于某件事 $x$ 发生的概率记作 $y=1$，则有 $p(y=0|x)=1-p$ 。假设向量 $x=(x_1,x_2,\cdots,x_k)^T$ 是由 $k$ 个变量组成的，那么逻辑回归模型就可以表示为：

$$P_i(y_i=1|x_i)=\frac{1}{1+e^{-g(x)}}$$

$y_i=0$ 表示某个事件没有发生， $y_i=1$ 表示某个事件确定发生。

逻辑回归的优点如下：

1. 实现简单，广泛的应用于工业问题上；
2. 分类时计算量非常小，速度很快，存储资源低；
3. 便利的观测样本概率分数；
4. 对逻辑回归而言，多重共线性并不是问题，它可以结合L2正则化来解决该问题；
5. 计算代价不高，易于理解和实现。

逻辑回归的缺点如下：

1. 当特征空间很大时，逻辑回归的性能不是很好；
2. 容易欠拟合，一般准确度不太高；
3. 不能很好地处理大量多类特征或变量；
4. 只能处理两分类问题，且必须线性可分；
5. 对于非线性特征，需要进行转换。

### 4.3.3 支持向量机（SVM）

支持向量机是一个二分类模型，寻求最优线性模型作为分类边界。应用领域包括文本分类和图像识别。超平面是分割输入变量空间的线。在SVM中，选择超平面以最佳地将输入变量空间中的点与它们的类（0级或1级）分开。在二维中，我们可以将其视为一条线，并假设我们的所有输入点都可以被这条线完全分开。SVM学习算法找到导致超平面最好地分离类的系数。

给定训练样本集 $D=(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)$ ， $y_i\in\{-1,+1\}$ 。在样本空间中，划分超平面可以通过如下线性方程来描述：

$$W^Tx+b=0$$

其中 $w=(w_1,w_2,\cdots,w_d)$ 为法向量，决定了超平面的方向，$b$ 为位移项，决定了超平面和原点之间的距离。显然，划分超平面可被法向量 $w$ 和位移 $b$ 确定，下面我们将其记为 $(w,b)$。样本中任意点 $x$ 到超平面 $(w,b)$ 的距离可写成：

$$r=\frac{|W^Tx+b|}{\|w\|}$$

假设超平面 $(w,b)$ 能够将训练样本正确分类，即对于 $(x_i,y_i)\in D$，若 $y_i=+1$，则有 $W^Tx_i+b>0$；若 $y_i=-1$，则有 $W^Tx_i+b<0$。令：

$$\begin{cases}W^Tx+b\geq+1, y_i=+1\\ W^Tx+b\leq-1, y_i=-1\end{cases}$$

如下图所示，距离超平面最近的这几个训练样本点使得上式的等号成立，它们被称为“支持向量”，这两个异类支持向量到超平面的距离之和为：

$$\Gamma=\frac{2}{\|w\|}$$

它们被称为间隔。

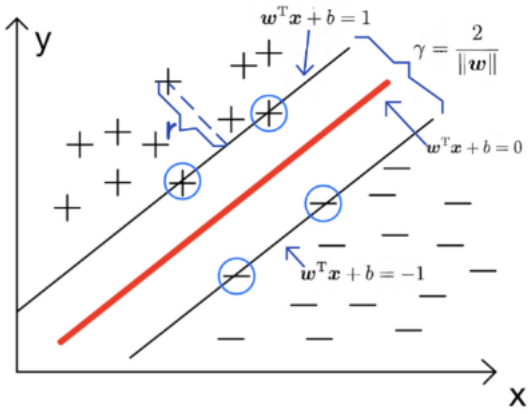


**图15：SVM的分离超平面**

欲找到具有“最大间隔”的划分超平面，也即是要找到能够满足上式中约束条件的参数 $w$ 和 $b$，使得 $\Gamma$ 最大化，即：

$$\max_{w,b}\frac{2}{\|w\|}$$

$$s.t.\quad y_i(W^Tx_i+b)\geq1, i=1,2,\cdots,m$$

SVM的实现步骤如下：

1. 计算误差：$E_i=f(x_i)-y_i=\sum_{j=1}^{n}\alpha_jy_jx_i^Tx_j+b-y_i$

2. 计算上界 $L$ 和下界 $H$：

$$\begin{cases} L = \max(0, \alpha_j^{old} - \alpha_i^{old}), H = \min(C, C + \alpha_j^{old} - \alpha_i^{old}) \; if \quad y_i \neq y_j \\ L = \max(0, \alpha_j^{old} + \alpha_i^{old} - C), H = \min(C, \alpha_j^{old} + \alpha_i^{old}) \; if \quad y_i = y_j \end{cases}$$

3. 计算eta： $\eta = x_i^T x_i + x_j^T x_j - 2x_i^T x_j$

4. 更新 $\alpha_j$： $\alpha_j^{new} = \alpha_j^{old} + \dfrac{y_j(E_i - E_j)}{\eta}$

5. 修剪 $\alpha_j$： $\alpha^{new,clipped} = \begin{cases} H & (\alpha_2^{new} \geq H) \\ \alpha_2^{new} & (L \leq \alpha_2^{new} \leq H) \\ L & \alpha_2^{new} \leq L \end{cases}$

6. 更新 $\alpha_i$： $\alpha_i^{new} = \alpha_i^{old} + y_i y_j (\alpha_j^{old} - \alpha_j^{new,clipped})$

7. 更新 $b_1$ 和 $b_2$

$$b_1^{new} = b^{old} - E_i - y_i(\alpha_i^{new} - \alpha_i^{old}) x_i^T x_i - y_j(\alpha_j^{new} - \alpha_j^{old}) x_j^T x_i$$
$$b_2^{new} = b^{old} - E_j - y_i(\alpha_i^{new} - \alpha_i^{old}) x_i^T x_j - y_j(\alpha_j^{new} - \alpha_j^{old}) x_j^T x_j$$

8. 更新 $b$： $b = \begin{cases} b_1 & 0 < \alpha_1^{new} < C \\ b_2 & 0 < \alpha_2^{new} < C \\ \dfrac{b_1 + b_2}{2} & otherwise \end{cases}$

SVM的优点如下：

1. 可以解决高维问题，即大型特征空间；
2. 解决小样本下机器学习问题；
3. 能够处理非线性特征的相互作用；
4. 无局部极小值问题；
5. 无需依赖整个数据；
6. 泛化能力比较强。

SVM的缺点如下：

1. 当观测样本很多时，效率并不是很高；
2. 对非线性问题没有通用解决方案，有时候很难找到一个合适的核函数；
3. 对于核函数的高维映射解释力不强，尤其是径向基函数；
4. 常规SVM只支持二分类；
5. 对缺失数据敏感。

#### 4.3.4 KNN算法

KNN算法的核心思想是如果一个样本在特征空间中的 $k$ 个最相邻的样本中的大多数属于某

一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。该方法在确定分类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。KNN方法在类别决策时，只与极少量的相邻样本有关。由于KNN方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于类域的交叉或重叠较多的待分样本集来说，KNN方法较其他方法更为适合。

KNN是一种惰性机器学习方法，其优点如下：

1. 天生支持增量学习（不需要训练，没有增量拓展的麻烦事儿）；
2. 可以用于非线性分类；
3. 能对超多变形的复杂决策空间建模；
4. 在数据量不多但数据代表性较强时，KNN分类效果较好。

缺点如下：

1. 计算开销大；
2. 可解释性不强；
3. 样本不平衡的时候，对稀有类别的预测准确率低。

### 4.3.5 决策树算法

决策树又称为判定树，是运用于分类的一种树结构，其中的每个内部节点代表对某一属性的一次测试，每条边代表一个测试结果，叶节点代表某个类或类的分布。决策树的决策过程就是从根节点开始，测试待分类项中对应的特征属性，并按照其值选择输出分支，直到叶子节点，将叶子节点的存放的类别作为决策果。

下面是常见的决策树的启发函数：

信息熵定义为：

$$Ent(D) = -\sum_{k=1}^{n} p_k \log_2 p_k$$

ID3决策树中的信息增益定义为：

$$Gain(D,a) = Ent(D) - Ent(D \mid a) = Ent(D) - \sum_{v=1}^{V} \frac{D_v}{D} Ent(D_v)$$

基尼值定义为：

$$Gini(D) = \sum_{k=1}^{|y|} \sum_{k \neq k} p_k p_k = 1 - \sum_{k=1}^{|y|} p_k^2$$

C4.5算法中定义的信息增益率为：

$$Gain\_ratio(D,a) = \frac{Gain(D,a)}{IV(a)}$$

CART决策树中的基尼指数定义为：

$$Gini\_index(D,a)=\sum_{v=1}^{V}\frac{D_v}{D}Gini(D_v)$$

与其他分类算法相比决策树有如下优点：

1. 速度快：计算量相对较小，且容易转化成分类规则。只要沿着树根向下一直走到叶，沿途的分裂条件就能够唯一确定一条分类的谓词；
2. 准确性高：挖掘出的分类规则准确性高，便于理解，决策树可以清晰的显示那些字段比较重要。

决策树的缺点：

1. 缺乏伸缩性：由于进行深度优先搜索，所以算法受内存大小限制，难以处理大训练集。
2. 为了处理大数据集或连续量的种种改进算法（离散化、取样）不仅增加了分类算法的额外开销，而且降低了分类的准确性，对连续性的字段比较难以预测，当类别太多时，错误可能就会增加的比较快，对有时间顺序的数据，需要很多预处理的工作。

### 4.3.6 随机森林算法

随机森林本质上属于机器学习的一大分支—集成学习，是将许多棵决策树整合成森林并用来预测最终结果的方法。

随机森林实际上是一种特殊的bagging方法，它将决策树用作bagging中的模型。首先，用bootstrap方法生成 $m$ 个训练集，然后，对于每个训练集，构造一颗决策树，在节点找特征进行分裂的时候，并不是对所有特征找到能使得指标（如信息增益）最大的，而是在特征中随机抽取一部分特征，在抽到的特征中间找到最优解，应用于节点，进行分裂。随机森林的方法由于有了bagging，也就是集成的思想，实际上相当于对于样本和特征都进行了采样，所以可以避免过拟合。预测阶段的方法就是bagging的策略：分类投票和回归均值。

有了树我们就可以分类了，但是森林中的每棵树是怎么生成的呢？每棵树的按照如下规则生成：

1. 如果训练集大小为 $N$，对于每棵树而言，随机且有放回地从训练集中的抽取 $N$ 个训练样本，作为该树的训练集；
2. 如果每个样本的特征维度为 $M$，指定一个常数 $m \le M$，随机地从 $M$ 个特征中选取 $m$ 个特征子集，每次树进行分裂时，从这 $m$ 个特征中选择最优的；
3. 每棵树都尽最大程度的生长，并且没有剪枝过程。

随机森林算法的优点如下：

1. 由于采用了集成算法，本身精度比大多数单个算法要好，所以准确性高在测试集上表现良好，由于两个随机性的引入，使得随机森林不容易陷入过拟合（样本随机，特征随机）；

2. 在工业上，由于两个随机性的引入，使得随机森林具有一定的抗噪声能力，对比其他算法具有一定优势；
3. 由于树的组合，使得随机森林可以处理非线性数据，本身属于非线性分类（拟合）模型；
4. 它能够处理很高维度（feature很多）的数据，并且不用做特征选择，对数据集的适应能力强：既能处理离散型数据，也能处理连续型数据，数据集无需规范化；
5. 训练速度快，可以运用在大规模数据集上；
6. 在训练过程中，能够检测到feature间的互相影响，且可以得出feature的重要性，具有一定参考意义。

随机森林的缺点为：

1. 当随机森林中的决策树个数很多时，训练时需要的空间和时间会比较大；
2. 随机森林中还有许多不好解释的地方，有点算是黑盒模型；
3. 在某些噪音比较大的样本集上，RF的模型容易陷入过拟合。

### 4.3.7 GBDT算法

GBDT的基本结构是决策树组成的森林，学习方式是梯度提升。

具体的讲，GBDT作为集成模型，预测的方式是把所有子树的结果加起来。GBDT通过逐一生成决策子树的方式生成整个森林，生成新子树的过程是利用样本标签值与当前树林预测值之间的残差，构建新的子树。

训练数据集$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$；函数的损失函数为$L(y,f(x))$。

算法的实现步骤为：

1. 初始化 $f_0(x)=0$；
2. 对 $m=1,2,\cdots,M$

   1)．极小化损失函数

   $$(\beta_m,\gamma_m)=\arg\min_{\beta,\gamma}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+\beta b(x_i,\gamma))$$

   2)．更新

   $$f_m(x)=f_{m-1}(x)+\beta_m b(x,\gamma_m)$$

3. 最终得到强学习模型 $f(x)$

   $$f(x)=f_M(x)=\sum_{m=1}^{M}\beta_m b(x,\gamma_m)$$

总之，提升方法告诉我们如何来求一个效果更好模型，那就是将多个弱模型组合起来。

GBDT主要的优点有：

1. 可以灵活处理各种类型的数据，包括连续值和离散值；

2. 在相对少的调参时间情况下，预测的准备率也可以比较高。这个是相对SVM来说的；
3. 使用一些健壮的损失函数，对异常值的鲁棒性非常强。比如Huber损失函数和Quantile损失函数；
4. 很好的利用了弱分类器进行级联；
5. 充分考虑的每个分类器的权重。

GBDT的主要缺点有：

1. 由于弱学习器之间存在依赖关系，难以并行训练数据。不过可以通过自采样的SGBT来达到部分并行。

### 4.3.8 LightGBM算法

GBDT是机器学习中的一个非常流行并且有效的算法模型，它是一个基于决策树的梯度提升算法。在Kaggle比赛中，XGBoost等基于GBDT思想的算法有着非常好的表现，据统计Kaggle比赛中，50%以上的冠军方案都是基于GBDT算法。但是大训练样本和高维度特征的数据环境下，GBDT算法的性能以及准确性却面临了极大的挑战。为了解决这些问题，LightGBM应运而生，引用LightGBM官网对该框架的介绍，LightGBM具有以下特点：

1. 更快的训练速度和效率；
2. 更低的内存使用；
3. 更好的准确率；
4. 支持并行化学习；
5. 可以处理大规模数据。

LightGBM算法的缺点如下：

1. 可能会长出比较深的决策树，产生过拟合。因此LightGBM在Leaf-wise之上增加了一个最大深度限制，在保证高效率的同时防止过拟合；
2. Boosting族是迭代算法，每一次迭代都根据上一次迭代的预测结果对样本进行权重调整，所以随着迭代不断进行，误差会越来越小，模型的偏差（bias）会不断降低，所以会对噪点较为敏感；
3. 在寻找最优解时，依据的是最优切分变量，没有将最优解是全部特征的综合这一理念考虑进去。

### 4.3.9 Xgboost算法

该算法思想就是不断地添加树，不断地进行特征分裂来生长一棵树，每次添加一个树，其实是学习一个新函数，去拟合上次预测的残差。当我们训练完成得到$k$棵树，我们要预测一个样本的分数，其实就是根据这个样本的特征，在每棵树中会落到对应的一个叶子节点，每个叶子节点就对应一个分数，最后只需要将每棵树对应的分数加起来就是该样本的预测值。

$$\tilde{y} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i)$$

$$where \quad F = \{ f(x) = w_{q(x)} \} (q : R^m \to T, w \in R^T)$$

注：$w_{q(x)}$ 为叶子节点 $q$ 的分数，$f(x)$ 为其中一棵树。

算法的优点为：

1. 使用许多策略去防止过拟合，如：正则化项、Shrinkage and Column Subsampling等；
2. 目标函数优化利用了损失函数关于待求函数的二阶导数；
3. 支持并行化，这是XGBoost的闪光点，虽然树与树之间是串行关系，但是同层级节点可并行。具体的对于某个节点，节点内选择最佳分裂点，候选分裂点计算增益用多线程并行。训练速度快；
4. 添加了对稀疏数据的处理；
5. 交叉验证，early stop，当预测结果已经很好的时候可以提前停止建树，加快训练速度；
6. 支持设置样本权重，该权重体现在一阶导数 $g$ 和二阶导数 $h$，通过调整权重可以去更加关注一些样本。

### 4.3.10 分类预测评估方法

模型评估的目标是选出泛化能力强的模型完成机器学习任务。实际的机器学习任务往往需要进行大量的实验，经过反复调参、使用多种模型算法（甚至多模型融合策略）来完成自己的机器学习问题，并观察哪种模型算法在什么样的参数下能够最好地完成任务。

在分类问题中，混淆矩阵是非常有效的评估模式，特别用于监督学习（在无监督学习中一般叫做匹配矩阵）。典型的混淆矩阵构成如下图所示：

**表10：混淆矩阵**

| | | Predicted | |
|---|---|---|---|
| Actual | | Positive | Negative |
| | Positive | TP | FN |
| | Negative | FP | TN |

其中，由混淆矩阵可得TP、FN、FP、TN相应的数值，进而求得模型的正确率与错误率。

Accuracy=(TP+TN)/(TP+TN+FP+FN)

Error Rate=1-Accuracy=(FP+FN)/(TP+TN+FP+FN)

准确率(Precision)：所有预测正类样本被正确分类的比率，即预测为正类的样本实际也为正类的比例。

Precision=TP/(TP+FP)

召回率(Recall)：被正确分类的样本在实际正类样本中所占的比率，也被称为敏感度(Sensitivity)。

Recall=TP/(TP+FN)

F测度（F measure）：基于准确率和召回率的衡量指标。（$\beta$常取1）

F measure=$(1+\beta)^2$ *Recall*Precision)/($\beta^2$ *Recall+Precision)

以上这些测量指标虽然相较正确率、错误率而言能够更好的衡量数据，但准确率未考虑预测的负类数据，召回率仅考虑实际的正类样本数据，因此考虑使用ROC曲线对模型进行进一步的评估。

目前在机器学习中广泛使用的评估方法是ROC曲线，ROC曲线是通过绘制TP率(Sensitivity)和FP率(Specificity)的关系得到衡量分类预测精度：

Specificity=TN/(TN+FP)

ROC曲线将收益和损失的信息直观的表现了出来，每一点表示了每个分类器在特定分布上的预测效果，而曲线下方的面积(AUC)越大，也表明分类的效果越好。在大多数情况下，ROC曲线都具有很好的评估能力。

# 5 模型的求解

## 5.1 Adaboost模型

### 5.1.1 模型的结果

参数默认的情况下，我们得到的准确率为0.9054951264302867，具体的评估结果如下：

**表11：Adaboost的模型评估**

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.88 | 0.94 | 0.91 | 7045 |
| 1 | 0.93 | 0.88 | 0.90 | 7113 |
| accuracy | | | 0.91 | 14158 |
| macro avg | 0.91 | 0.91 | 0.91 | 14158 |
| weighted avg | 0.91 | 0.91 | 0.91 | 14158 |

我们利用网格搜索选取最佳参数为：{'learning_rate': 1, 'n_estimators': 500}，此时的评估结果为：

**表12：调参后的Adaboost的模型评估**

| | Precision | recall | f1-score | support |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 0 | 0.90 | 0.98 | 0.94 | 7045 |
| 1 | 0.97 | 0.89 | 0.93 | 7113 |
| accuracy | | | 0.93 | 14158 |
| macro avg | 0.94 | 0.93 | 0.93 | 14158 |
| weighted avg | 0.94 | 0.93 | 0.93 | 14158 |

我们用调参后的模型训练，准确率为0.934595281819466。

此时五折交叉验证的结果如下：

表13：五折交叉验证结果

| | | | | |
|---|---|---|---|---|
| 0.93385803 | 0.93264719 | 0.93764189 | 0.93416074 | 0.93536179 |

我们进一步绘制ROC曲线如下：

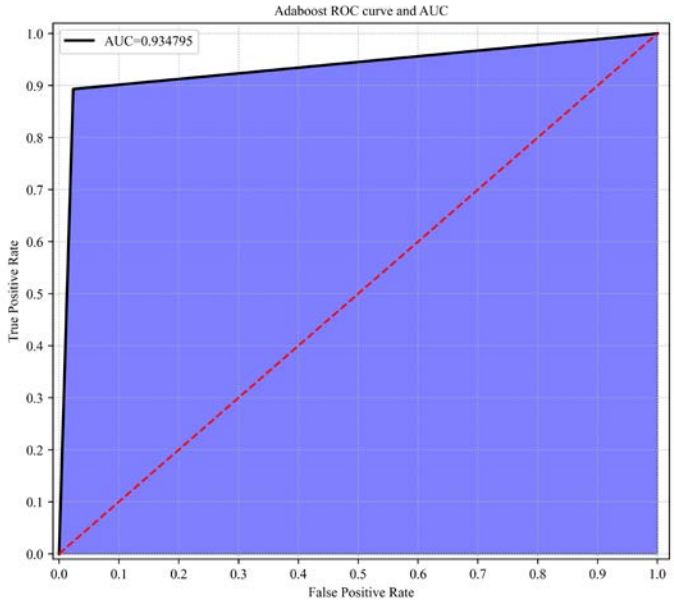


图16：Adaboost模型的ROC曲线

通过调参后的结果可知，训练模型的正确率达到了93%，正类样本中有89%分类正确，负类样本中有98%被分类正确，F值为0.93，得到的AUC的值为0.93475。虽然样本不均衡带来的问题并没有完全消弭，但是依旧可以认为模型不错。

### 5.1.2 模型的混淆矩阵

Accuracy: 0.934595281819466

Precision: 0.9743904309155037

Recall: 0.8932939687895403

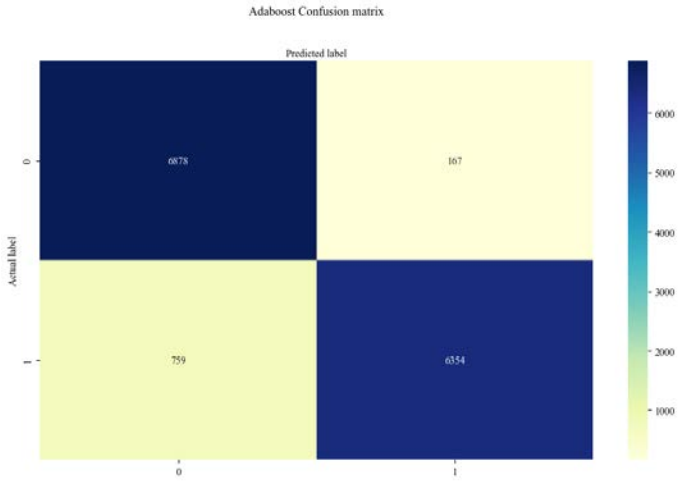


图17：Adaboost模型的混淆矩阵

## 5.2 Logistic 回归模型

### 5.2.1 模型的结果

参数默认的情况下，我们得到的准确率为0.9357253849413759，具体的评估结果如下：

表14：Logistic 回归的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.98 | 0.94 | 7045 |
| 1 | 0.98 | 0.89 | 0.93 | 7113 |
| accuracy | | | 0.94 | 14158 |
| macro avg | 0.94 | 0.94 | 0.94 | 14158 |
| weighted avg | 0.94 | 0.94 | 0.94 | 14158 |

我们利用网格搜索选取最佳参数为：{'C': 10, 'penalty': 'l1', 'solver': 'liblinear'}，此时的评估结果为：

表15：调参后的Logistic 回归的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.98 | 0.94 | 7045 |
| 1 | 0.98 | 0.89 | 0.93 | 7113 |
| accuracy | | | 0.94 | 14158 |
| macro avg | 0.94 | 0.94 | 0.94 | 14158 |
| weighted avg | 0.94 | 0.94 | 0.94 | 14158 |

可以看出，调参前后的准确率没有太大的提高，调参后的准确率为0.9357960163864952。

此时五折交叉验证的结果如下：

表16：五折交叉验证结果

| | | | | |
|---|---|---|---|---|
| 0.93567428 | 0.93416074 | 0.93870138 | 0.93703648 | 0.93808659 |

我们进一步绘制ROC曲线如下：

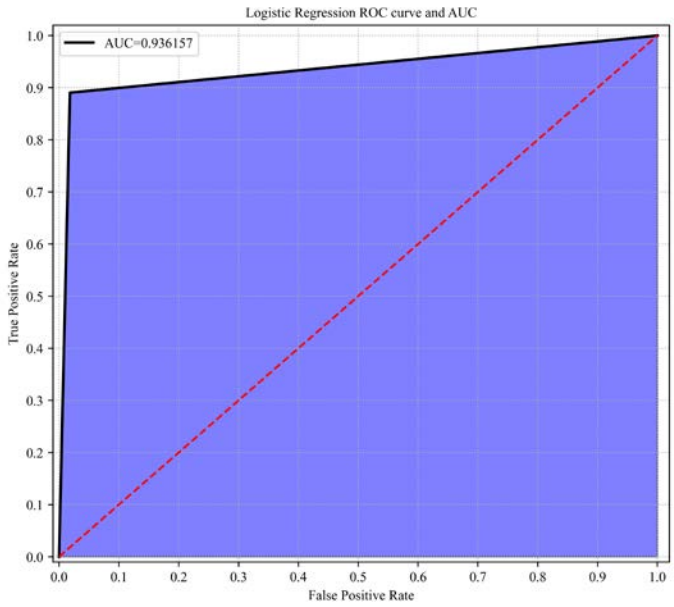


图18：Logistic回归模型的ROC曲线

通过调参后的结果可知，训练模型的正确率达到了94%，正类样本中有89%分类正确，负类样本中有98%被分类正确，F值为0.94，得到的AUC的值为0.93617。

### 5.2.2 模型的混淆矩阵

Accuracy：0.935937279276734

Precision：0.9801918910554008

Recall：0.8904822156614649

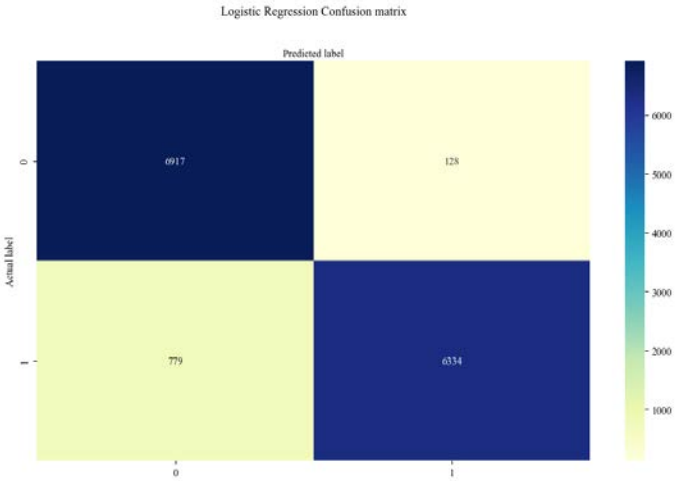


图19：Logistic回归模型的混淆矩阵

## 5.3 支持向量机模型

### 5.3.1 模型的结果

参数默认的情况下，我们得到的准确率为0.8946885153270235，具体的评估结果如下：

表17：支持向量机的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.86 | 0.94 | 0.90 | 7045 |
| 1 | 0.94 | 0.85 | 0.89 | 7113 |
| accuracy | | | 0.89 | 14158 |
| macro avg | 0.90 | 0.89 | 0.89 | 14158 |
| weighted avg | 0.90 | 0.89 | 0.89 | 14158 |

我们利用网格搜索选取最佳参数为：{'C': 1.0, 'gamma': 0.01}，此时的评估结果为：

表18：调参后的支持向量机的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.89 | 0.97 | 0.93 | 7045 |
| 1 | 0.97 | 0.89 | 0.92 | 7113 |
| accuracy | | | 0.93 | 14158 |
| macro avg | 0.93 | 0.93 | 0.93 | 14158 |
| weighted avg | 0.93 | 0.93 | 0.93 | 14158 |

调参后的模型准确率为0.9273908744172906。

五折交叉验证的结果如下：

表19：五折交叉验证结果

| | | | | |
|---|---|---|---|---|
| 0.92417133 | 0.92477675 | 0.92992281 | 0.9229605 | 0.92537087 |

ROC曲线如下所示：

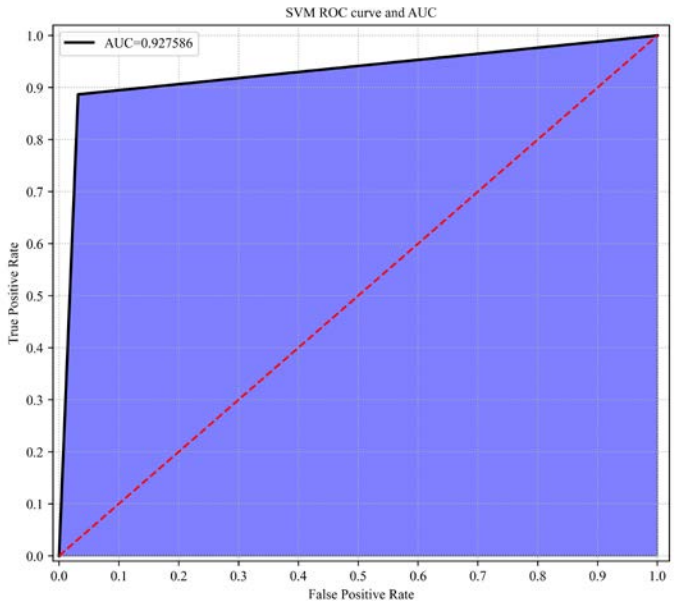


**图20：支持向量机模型的ROC曲线**

通过调参后的结果可知，训练模型的正确率达到了93%，正类样本中有89%分类正确，负类样本中有97%被分类正确，F值为0.93，得到的AUC的值为0.93。

### 5.3.2 模型的混淆矩阵

Accuracy：0.9273908744172906

Precision：0.9657125363538956

Recall：0.8869675242513707

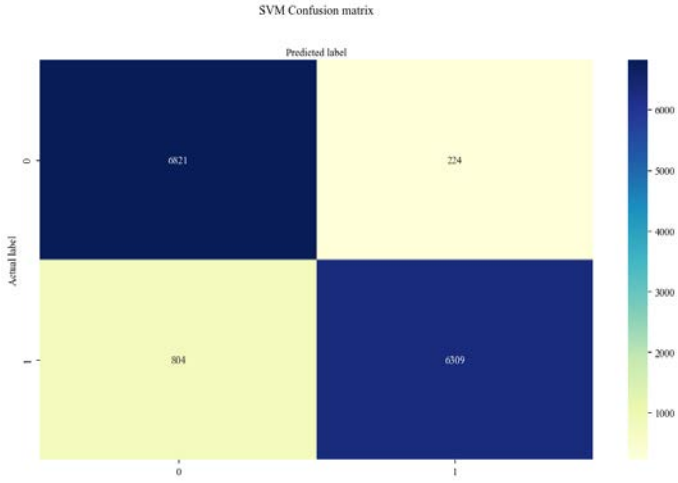


**图21：支持向量机模型的混淆矩阵**

## 5.4 KNN模型

### 5.4.1 模型的结果

参数默认的情况下，我们得到的准确率为0.8689786693035739，具体的评估结果如下：

**表20：KNN的模型评估**

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 0.76 | 0.85 | 7045 |

| | | | | |
|---|---|---|---|---|
| 1 | 0.80 | 0.98 | 0.88 | 7113 |
| accuracy | | | 0.87 | 14158 |
| macro avg | 0.89 | 0.87 | 0.87 | 14158 |
| weighted avg | 0.89 | 0.87 | 0.87 | 14158 |

我们利用网格搜索选取最佳参数为：{'n_neighbors': 5, 'p': 2, 'weights': 'distance'}，此时的评估结果为：

**表20：调参后的KNN的模型评估**

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 0.77 | 0.86 | 7045 |
| 1 | 0.81 | 0.98 | 0.89 | 7113 |
| accuracy | | | 0.87 | 14158 |
| macro avg | 0.89 | 0.87 | 0.87 | 14158 |
| weighted avg | 0.89 | 0.87 | 0.87 | 14158 |

综上可知，调参后的模型的准确率为0.8744879220228846。

五折交叉验证的结果如下：

**表21：五折交叉验证结果**

| | | | | |
|---|---|---|---|---|
| 0.85893749 | 0.85485092 | 0.86347813 | 0.86060239 | 0.85195277 |

ROC曲线如下所示：

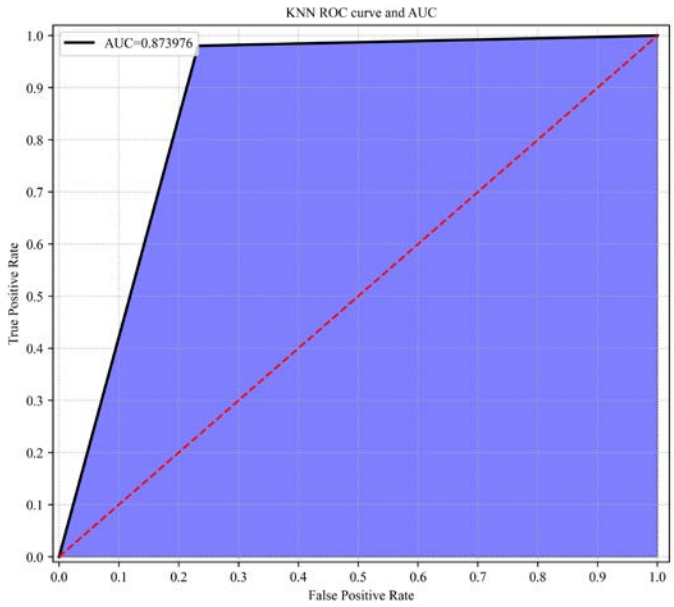


**图22：KNN模型的ROC曲线**

通过调参后的结果可知，训练模型的正确率达到了87%，正类样本中有98%分类正确，负类

样本中有77%被分类正确，F值为0.87，得到的AUC的值为0.87。

### 5.4.2 模型的混淆矩阵

Accuracy：0.8744879220228846

Precision：0.8098002786809103

Recall：0.9804583157598763

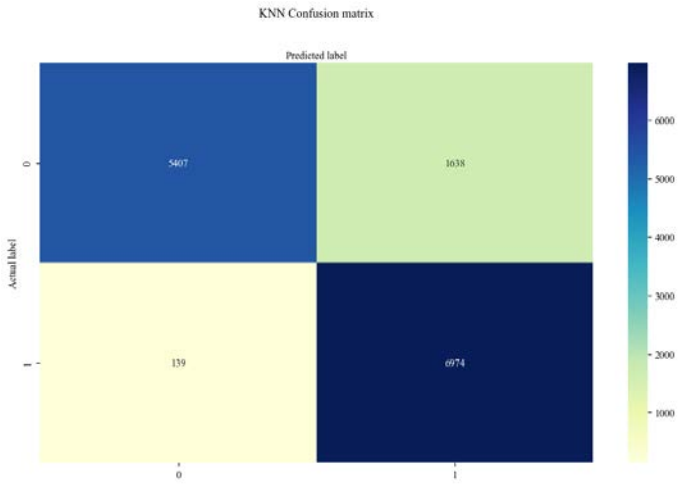


图23：KNN模型的混淆矩阵

## 5.5 决策树模型

### 5.5.1 模型的结果

参数默认的情况下，我们得到的准确率为0.9009747139426473，具体的评估结果如下：

表22：决策树的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.89 | 0.90 | 7045 |
| 1 | 0.89 | 0.91 | 0.90 | 7113 |
| accuracy | | | 0.90 | 14158 |
| macro avg | 0.90 | 0.90 | 0.90 | 14158 |
| weighted avg | 0.90 | 0.90 | 0.90 | 14158 |

我们利用网格搜索选取最佳参数为：{'criterion': 'gini', 'max_depth': 15}，此时的评估结果为：

表23：调参后的决策树的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.89 | 0.92 | 0.90 | 7045 |
| 1 | 0.92 | 0.88 | 0.90 | 7113 |

| | | | | |
|---|---|---|---|---|
| accuracy | | | 0.90 | 14158 |
| macro avg | 0.90 | 0.90 | 0.90 | 14158 |
| weighted avg | 0.90 | 0.90 | 0.90 | 14158 |

此时模型的准确率为0.9011866082780053。

五折交叉验证的结果如下：

表24：五折交叉验证结果

| | | | | |
|---|---|---|---|---|
| 0.89617073 | 0.9019222 | 0.90283033 | 0.88981383 | 0.89872843 |

ROC曲线如下所示：

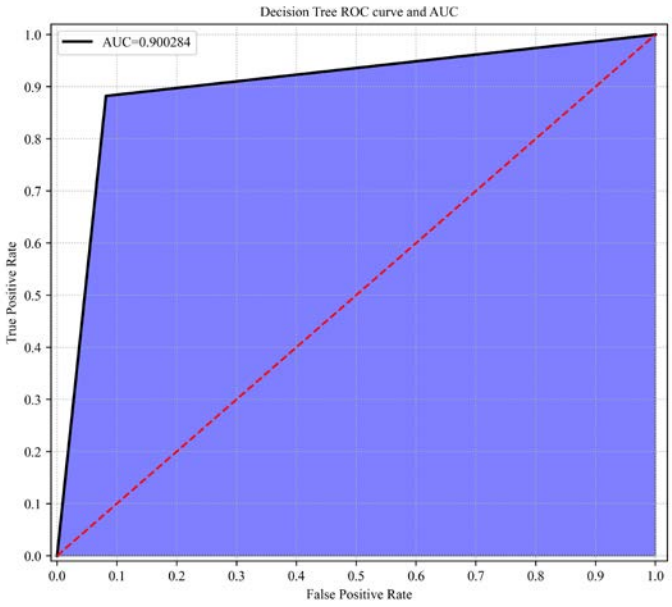


图24：决策树模型的ROC曲线

通过调参后的结果可知，训练模型的正确率达到了90%，正类样本中有88%分类正确，负类样本中有92%被分类正确，F值为0.90，得到的AUC的值为0.90。

### 5.5.2 模型的混淆矩阵

Accuracy：0.9011866082780053

Precision：0.9186694021101993

Recall：0.88134401799522

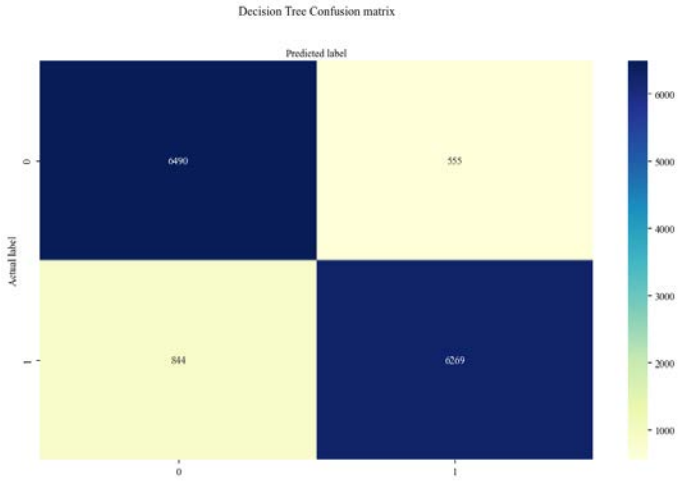


图25：决策树模型的混淆矩阵

## 5.6 随机森林模型

### 5.6.1 模型的结果

参数默认的情况下，我们得到的准确率为0.9398926402034186，具体的评估结果如下：

表25：随机森林的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.95 | 0.94 | 7045 |
| 1 | 0.95 | 0.93 | 0.94 | 7113 |
| accuracy | | | 0.94 | 14158 |
| macro avg | 0.94 | 0.94 | 0.94 | 14158 |
| weighted avg | 0.94 | 0.94 | 0.94 | 14158 |

我们利用网格搜索选取最佳参数为：{'criterion': 'entropy', 'max_depth': 19, 'n_estimators': 500}，此时的评估结果为：

表26：调参后的随机森林的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.95 | 0.94 | 7045 |
| 1 | 0.95 | 0.93 | 0.94 | 7113 |
| accuracy | | | 0.94 | 14158 |
| macro avg | 0.94 | 0.94 | 0.94 | 14158 |
| weighted avg | 0.94 | 0.94 | 0.94 | 14158 |

调参后的模型准确率为0.9373499081791213。

参数默认状况下的五折交叉验证的结果如下：

表27：五折交叉验证结果

| | | | | |
|---|---|---|---|---|
| 0.93537158 | 0.93325261 | 0.93870138 | 0.93400938 | 0.932637 |

ROC曲线如下所示：

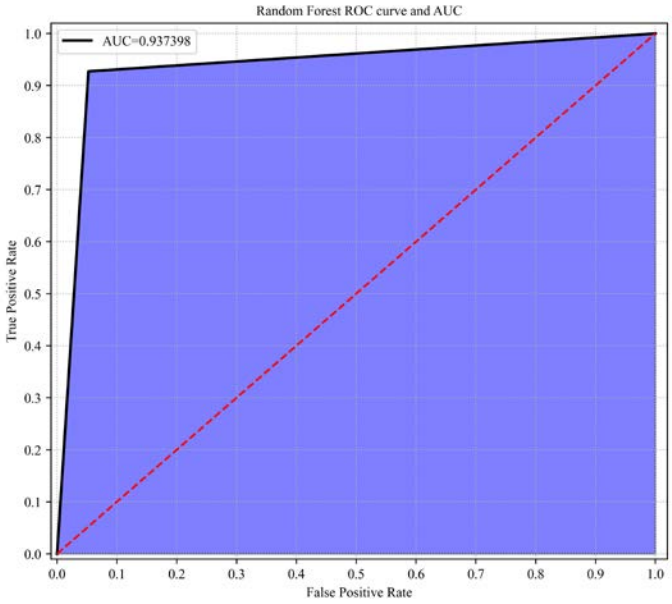


图26：随机森林模型的ROC曲线

结果可知，训练模型的正确率达到了94%，正类样本中有93%分类正确，负类样本中有95%被分类正确，F值为0.94，得到的AUC的值为0.937。

### 5.6.2 模型的结果

Accuracy：0.9373499081791213

Precision：0.9468848693654895

Recall：0.9273161816392521

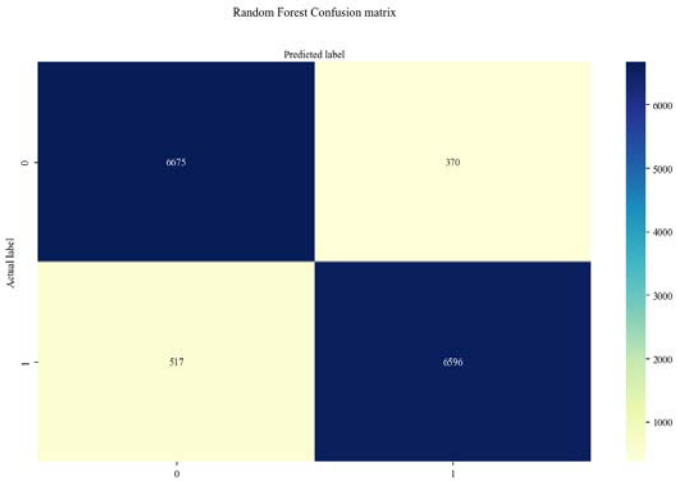


图27：随机森林模型的混淆矩阵

## 5.7 GBDT模型

### 5.7.1 模型的结果

参数默认的情况下，我们得到的准确率为0.9143240570702077，具体的评估结果如下：

表28：GBDT的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.89 | 0.95 | 0.92 | 7045 |
| 1 | 0.95 | 0.88 | 0.91 | 7113 |
| accuracy | | | 0.91 | 14158 |
| macro avg | 0.92 | 0.91 | 0.91 | 14158 |
| weighted avg | 0.92 | 0.91 | 0.91 | 14158 |

我们利用网格搜索选取最佳参数为：{'n_estimators': 500}，此时的评估结果为：

表29：调参后的GBDT的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.97 | 0.94 | 7045 |
| 1 | 0.97 | 0.90 | 0.93 | 7113 |
| accuracy | | | 0.94 | 14158 |
| macro avg | 0.94 | 0.94 | 0.94 | 14158 |
| weighted avg | 0.94 | 0.94 | 0.94 | 14158 |

调参后的模型准确率为0.9352309648255404。

五折交叉验证的结果如下：

表30：五折交叉验证结果

| | | | | |
|---|---|---|---|---|
| 0.93446345 | 0.92992281 | 0.93839867 | 0.93612835 | 0.93188011 |

ROC曲线如下所示：

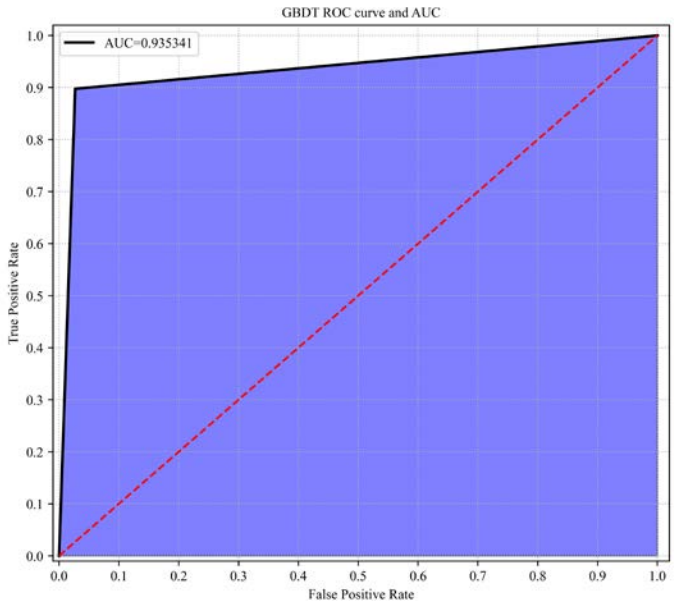


图28：GBDT模型的ROC曲线

结果可知，训练模型的正确率达到了94%，正类样本中有90%分类正确，负类样本中有97%被分类正确，F值为0.94，得到的AUC的值为0.935。

### 5.7.2 模型的混淆矩阵

Accuracy：0.9351603333804209

Precision：0.9711026615969581

Recall：0.8976521861380571

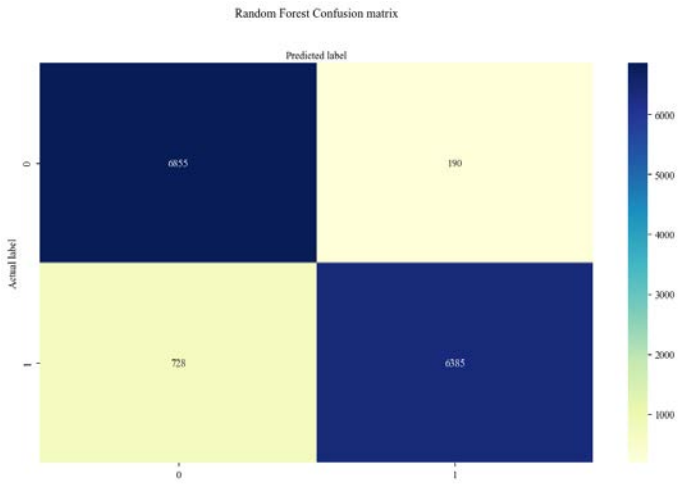


图29：GBDT模型的混淆矩阵

## 5.8 LightGBM模型

### 5.8.1 模型的结果

参数默认的情况下，我们得到的准确率为0.9143240570702077，具体的评估结果如下：

表31：LightGBM的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.93 | 0.92 | 7045 |

| | | | | |
|---|---|---|---|---|
| 1 | 0.93 | 0.91 | 0.92 | 7113 |
| accuracy | | | 0.92 | 14158 |
| macro avg | 0.92 | 0.92 | 0.92 | 14158 |
| weighted avg | 0.92 | 0.92 | 0.92 | 14158 |

我们利用网格搜索选取最佳参数为：{'max_depth': 13, 'n_estimators': 390}，此时的评估结果为：

表32：调参后的LightGBM的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.97 | 0.94 | 7045 |
| 1 | 0.97 | 0.91 | 0.94 | 7113 |
| accuracy | | | 0.94 | 14158 |
| macro avg | 0.94 | 0.94 | 0.94 | 14158 |
| weighted avg | 0.94 | 0.94 | 0.94 | 14158 |

调参后的模型准确率为0.9393275886424636，非常优秀！

五折交叉验证的结果如下：

表33：五折交叉验证结果

| | | | | |
|---|---|---|---|---|
| 0.93824731 | 0.93718783 | 0.94339337 | 0.94051763 | 0.93960036 |

ROC曲线如下所示：

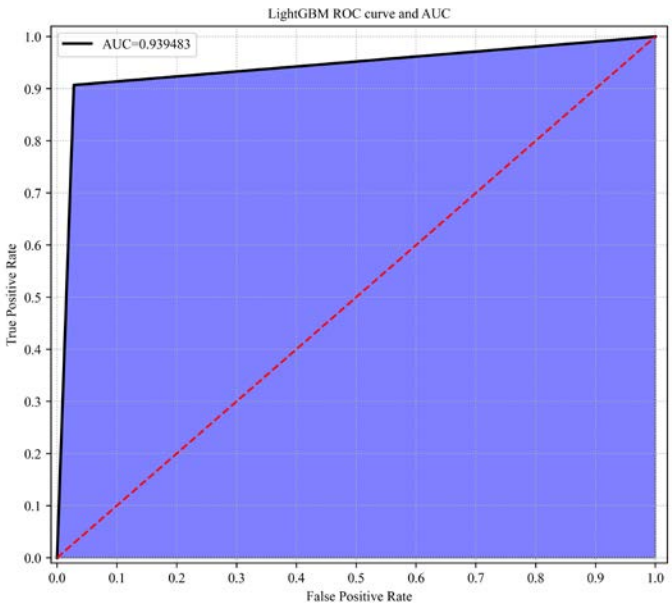


图30：LightGBM模型的ROC曲线

结果可知，训练模型的正确率达到了94%，正类样本中有91%分类正确，负类样本中有97%被

分类正确，F值为0.94，得到的AUC的值为0.939。

### 5.8.2 模型的混淆矩阵

Accuracy：0.9393275886424636

Precision：0.9702255639097744

Recall：0.9070715591171096

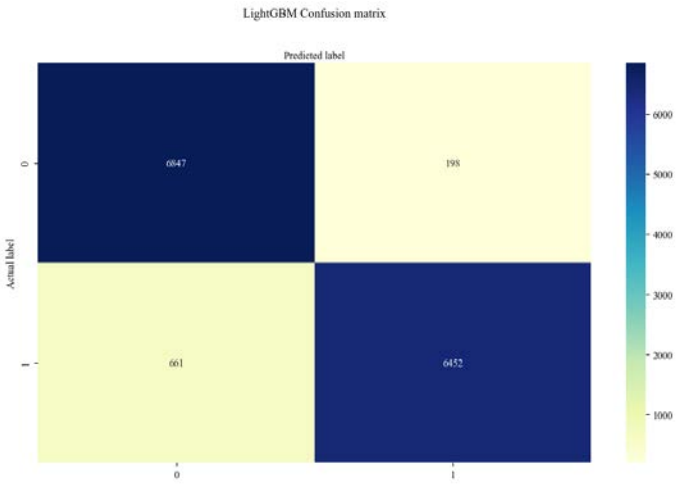


图31：LightGBM模型的混淆矩阵

## 5.9 Xgboost模型

### 5.9.1 模型的结果

参数默认的情况下，我们得到的准确率为0.9384093798559119，具体的评估结果如下：

表34：Xgboost的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.97 | 0.94 | 7045 |
| 1 | 0.97 | 0.91 | 0.94 | 7113 |
| accuracy | | | 0.94 | 14158 |
| macro avg | 0.94 | 0.94 | 0.94 | 14158 |
| weighted avg | 0.94 | 0.94 | 0.94 | 14158 |

我们利用网格搜索选取最佳参数为：{'max_depth'：8，'n_estimators'：100}，此时的评估结果为：

表35：调参后的Xgboost的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.97 | 0.94 | 7045 |
| 1 | 0.97 | 0.91 | 0.94 | 7113 |

| | | | | |
|---|---|---|---|---|
| accuracy | | | 0.94 | 14158 |
| macro avg | 0.94 | 0.94 | 0.94 | 14158 |
| weighted avg | 0.94 | 0.94 | 0.94 | 14158 |

这里调参的模型准确率为0.938833168526628，模型很优秀！

五折交叉验证结果如下：

表36：五折交叉验证结果

| | | | | |
|---|---|---|---|---|
| 0.93673377 | 0.93703648 | 0.94233389 | 0.94021492 | 0.93838934 |

ROC曲线如下所示：

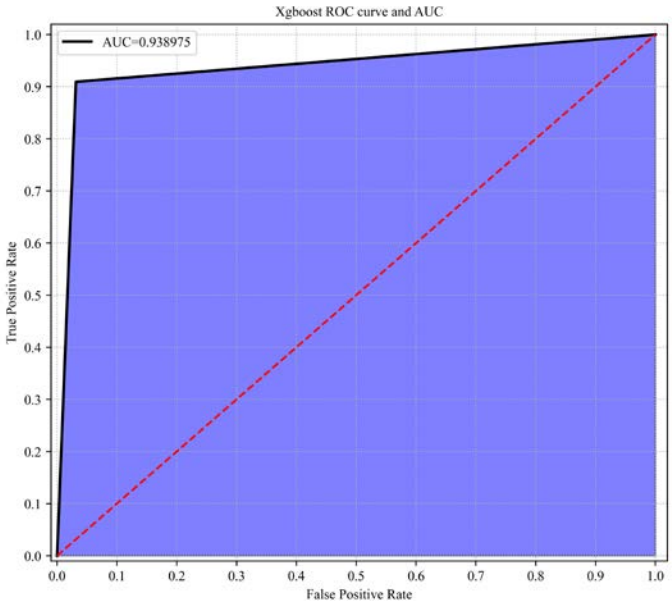


图32：Xgboost模型的ROC曲线

结果可知，训练模型的正确率达到了94%，正类样本中有91%分类正确，负类样本中有97%被分类正确，F值为0.94，得到的AUC的值为0.94。

### 5.9.2 模型的混淆矩阵

Accuracy：0.938833168526628

Precision：0.9668211029741444

Recall：0.9094615492759736

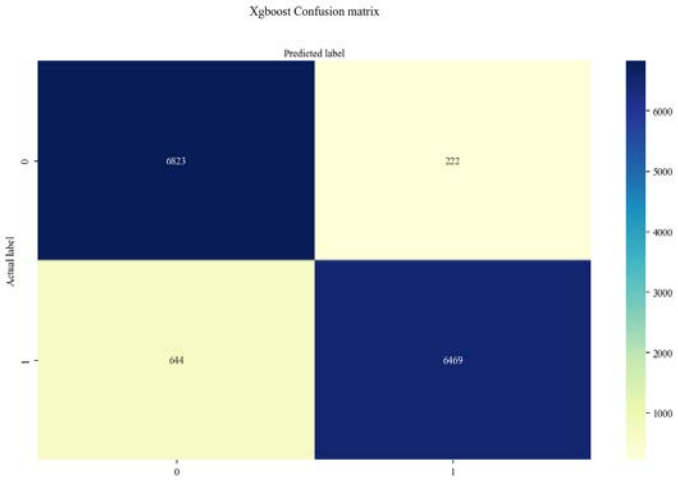


图33：Xgboost模型的混淆矩阵

# 6 模型的汇总

我们对29个常用的分类算法模型的准确率等评价指标进行汇总整理，结果如下：

表37：模型汇总

| Model | Accuracy | Balanced Accuracy | ROC AUC | F1 Score | Time Taken |
|---|---|---|---|---|---|
| RandomForestClassifier | 0.94 | 0.94 | 0.94 | 0.94 | 2.72 |
| LGBMClassifier | 0.94 | 0.94 | 0.94 | 0.94 | 0.33 |
| XGBClassifier | 0.94 | 0.94 | 0.94 | 0.94 | 0.81 |
| SVC | 0.94 | 0.94 | 0.94 | 0.94 | 32.54 |
| LinearSVC | 0.94 | 0.94 | 0.94 | 0.94 | 4.18 |
| CalibratedClassifierCV | 0.94 | 0.94 | 0.94 | 0.94 | 15.10 |
| LogisticRegression | 0.94 | 0.94 | 0.94 | 0.94 | 0.25 |
| SGDClassifier | 0.93 | 0.93 | 0.93 | 0.93 | 0.22 |
| ExtraTreesClassifier | 0.93 | 0.93 | 0.93 | 0.93 | 2.72 |
| BaggingClassifier | 0.93 | 0.93 | 0.93 | 0.93 | 1.27 |
| QuadraticDiscriminantAnalysis | 0.93 | 0.93 | 0.93 | 0.93 | 0.17 |
| KNeighborsClassifier | 0.92 | 0.92 | 0.92 | 0.92 | 0.94 |
| Perceptron | 0.92 | 0.92 | 0.92 | 0.92 | 0.11 |
| PassiveAggressiveClassifier | 0.92 | 0.92 | 0.92 | 0.92 | 0.12 |
| RidgeClassifier | 0.91 | 0.91 | 0.91 | 0.91 | 0.11 |
| RidgeClassifierCV | 0.91 | 0.91 | 0.91 | 0.91 | 0.19 |
| LinearDiscriminantAnalysis | 0.91 | 0.91 | 0.91 | 0.91 | 0.33 |
| AdaBoostClassifier | 0.91 | 0.91 | 0.91 | 0.91 | 1.48 |

| | | | | | |
|---|---|---|---|---|---|
| DecisionTreeClassifier | 0.90 | 0.90 | 0.90 | 0.90 | 0.22 |
| NuSVC | 0.90 | 0.90 | 0.90 | 0.90 | 60.39 |
| ExtraTreeClassifier | 0.90 | 0.90 | 0.90 | 0.90 | 0.11 |
| GaussianNB | 0.81 | 0.81 | 0.81 | 0.81 | 0.11 |
| NearestCentroid | 0.76 | 0.76 | 0.76 | 0.76 | 0.09 |
| BernoulliNB | 0.75 | 0.75 | 0.75 | 0.75 | 0.11 |
| DummyClassifier | 0.50 | 0.50 | 0.50 | 0.33 | 0.08 |

综上所述，RandomForestClassifier、LGBMClassifier、XGBClassifier、SVC、LinearSVC、CalibratedClassifierCV和LogisticRegression的准确率均为0.94，结合所需的时间，我们选择LGBMClassifier算法。

# 7 模型的预测

我们对新数据集进行预测，首先对数据集进行编码，对'campaign'、'passed_days'、'previous'、'emp_rate'、'cpi'、'cci'、'r3m'及'employed'列进行标准化。

展示一下标准化后的数据集：

| | age | campaign | passed_days | previous | emp_rate | cpi | cci | r3m | employed | job_admin. | ... | month_oct | month_sep | day_of_week_fri | day_of_week_mc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 48 | -0.60 | 0.20 | -0.33 | 0.85 | -0.24 | 0.93 | 0.79 | 0.85 | 0 | ... | 0 | 0 | 0 | |
| 1 | 46 | -0.21 | 0.20 | -0.33 | 0.66 | 0.70 | 0.87 | 0.72 | 0.35 | 0 | ... | 0 | 0 | 0 | |
| 2 | 43 | 2.92 | 0.20 | -0.33 | 0.85 | 0.57 | -0.47 | 0.78 | 0.85 | 1 | ... | 0 | 0 | 0 | |
| 3 | 49 | -0.60 | 0.20 | -0.33 | 0.85 | 0.57 | -0.47 | 0.78 | 0.85 | 0 | ... | 0 | 0 | 0 | |
| 4 | 39 | 4.48 | 0.20 | -0.33 | 0.85 | -0.24 | 0.93 | 0.79 | 0.85 | 0 | ... | 0 | 0 | 0 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 1995 | 34 | 2.13 | 0.20 | -0.33 | 0.85 | 1.50 | -0.28 | 0.73 | 0.85 | 0 | ... | 0 | 0 | 0 | |
| 1996 | 34 | -0.60 | 0.20 | -0.33 | -1.87 | -2.35 | 1.93 | -1.54 | -1.20 | 0 | ... | 0 | 0 | 0 | |
| 1997 | 25 | 3.70 | 0.20 | -0.33 | 0.85 | 0.57 | -0.47 | 0.79 | 0.85 | 0 | ... | 0 | 0 | 0 | |
| 1998 | 30 | 1.35 | 0.20 | -0.33 | 0.85 | -0.24 | 0.93 | 0.79 | 0.85 | 0 | ... | 0 | 0 | 0 | |
| 1999 | 28 | 2.52 | 0.20 | -0.33 | 0.85 | 0.57 | -0.47 | 0.79 | 0.85 | 0 | ... | 0 | 0 | 0 | |

2000 rows × 61 columns

图34：标准化后的数据集

我们看到这里编码和标准化后的数据有61列，而在上面我们训练时有62列。

我们看一下所有的列名：

```
Index(['age', 'campaign', 'passed_days', 'previous', 'emp_rate', 'cpi', 'cci',
       'r3m', 'employed', 'job_admin.', 'job_blue-collar', 'job_entrepreneur',
       'job_housemaid', 'job_management', 'job_retired', 'job_self-employed',
       'job_services', 'job_student', 'job_technician', 'job_unemployed',
       'job_unknown', 'marital_divorced', 'marital_married', 'marital_single',
       'marital_unknown', 'education_basic.4y', 'education_basic.6y',
       'education_basic.9y', 'education_high.school', 'education_illiterate',
       'education_professional.course', 'education_university.degree',
       'education_unknown', 'default_no', 'default_unknown', 'housing_no',
       'housing_unknown', 'housing_yes', 'loan_no', 'loan_unknown', 'loan_yes',
       'contact_cellular', 'contact_telephone', 'month_apr', 'month_aug',
       'month_dec', 'month_jul', 'month_jun', 'month_mar', 'month_may',
       'month_nov', 'month_oct', 'month_sep', 'day_of_week_fri',
       'day_of_week_mon', 'day_of_week_thu', 'day_of_week_tue',
       'day_of_week_wed', 'pre_outcome_failure', 'pre_outcome_nonexistent',
       'pre_outcome_success'],
      dtype='object')
```

图35：用于预测的数据集的列名

对比一下，我们发现缺少了'default_yes'列，我们使用选定的模型，去除这一列重新训练，得到的模型评价为：

表38：去除一列后的模型评估

| | Precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.91 | 0.97 | 0.94 | 7045 |
| 1 | 0.97 | 0.91 | 0.94 | 7113 |
| accuracy | | | 0.94 | 14158 |
| macro avg | 0.94 | 0.94 | 0.94 | 14158 |
| weighted avg | 0.94 | 0.94 | 0.94 | 14158 |

可以看出我们的F值为0.94，模型效果不错！

我们利用上面的模型进行预测，展示部分结果如下：

表39：部分预测结果展示

| ID | y |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| … | … |
| 1995 | 0 |
| 1996 | 1 |
| 1997 | 0 |
| 1998 | 0 |
| 1999 | 0 |

我们把1和0替换成“yes”和“no”，并添加到最后的数据集中，导入数据集。

我们来看看生成的数据集为：

| age | job | marital | education | default | housing | loan | contact | month | day_of_we | campaign | passed_da | previous | pre_outcor | emp_rate | cpi | cci | r3m | employed | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 48 | technician | married | high.schoc | unknown | no | yes | cellular | aug | tue | 1 | 999 | 0 | nonexisten | 1.4 | 93.444 | -36.1 | 4.963 | 5228.1 | no |
| 46 | blue-collar | married | profession | unknown | no | yes | telephone | may | tue | 2 | 999 | 0 | nonexisten | 1.1 | 93.994 | -36.4 | 4.856 | 5191 | no |
| 43 | admin. | married | high.schoc | no | yes | no | telephone | jul | mon | 10 | 999 | 0 | nonexisten | 1.4 | 93.918 | -42.7 | 4.96 | 5228.1 | no |
| 49 | services | married | high.schoc | unknown | yes | no | cellular | jul | wed | 1 | 999 | 0 | nonexisten | 1.4 | 93.918 | -42.7 | 4.957 | 5228.1 | no |
| 39 | technician | married | profession | no | no | no | cellular | aug | wed | 14 | 999 | 0 | nonexisten | 1.4 | 93.444 | -36.1 | 4.964 | 5228.1 | no |
| 37 | technician | single | profession | no | yes | no | cellular | aug | thu | 2 | 999 | 0 | nonexisten | 1.4 | 93.444 | -36.1 | 4.964 | 5228.1 | no |
| 30 | blue-collar | single | basic.9y | no | no | no | cellular | may | fri | 2 | 999 | 0 | nonexisten | -1.8 | 92.893 | -46.2 | 1.25 | 5099.1 | no |
| 29 | admin. | single | university.c | unknown | yes | no | cellular | aug | fri | 1 | 999 | 0 | nonexisten | 1.4 | 93.444 | -36.1 | 4.966 | 5228.1 | no |
| 50 | services | married | high.schoc | unknown | no | no | cellular | may | mon | 2 | 999 | 1 | failure | -1.8 | 92.893 | -46.2 | 1.299 | 5099.1 | no |
| 47 | unknown | married | unknown | unknown | no | no | telephone | jun | mon | 3 | 999 | 0 | nonexisten | 1.4 | 94.465 | -41.8 | 4.96 | 5228.1 | no |
| 44 | blue-collar | married | basic.4y | no | yes | no | cellular | jul | wed | 1 | 999 | 0 | nonexisten | 1.4 | 93.918 | -42.7 | 4.957 | 5228.1 | no |
| 73 | retired | married | basic.4y | no | no | no | cellular | aug | mon | 1 | 999 | 0 | nonexisten | -2.9 | 92.201 | -31.4 | 0.861 | 5076.2 | yes |
| 30 | unemploy | single | profession | no | yes | no | cellular | mar | mon | 1 | 999 | 0 | nonexisten | -1.8 | 93.369 | -34.8 | 0.646 | 5008.7 | yes |
| 44 | blue-collar | divorced | basic.4y | no | no | no | telephone | jul | wed | 1 | 999 | 0 | nonexisten | -1.7 | 94.215 | -40.3 | 0.84 | 4991.6 | yes |
| 42 | admin. | married | university.c | no | no | no | cellular | jul | wed | 2 | 999 | 0 | nonexisten | 1.4 | 93.918 | -42.7 | 4.963 | 5228.1 | no |
| 73 | retired | married | profession | no | yes | no | cellular | nov | fri | 1 | 999 | 0 | nonexisten | -1.1 | 94.767 | -50.8 | 1.028 | 4963.6 | no |
| 58 | services | married | basic.4y | no | no | no | telephone | may | thu | 5 | 999 | 0 | nonexisten | 1.1 | 93.994 | -36.4 | 4.855 | 5191 | no |
| 41 | blue-collar | married | basic.6y | unknown | no | no | telephone | may | fri | 4 | 999 | 0 | nonexisten | 1.1 | 93.994 | -36.4 | 4.855 | 5191 | no |
| 42 | technician | married | basic.9y | no | yes | no | telephone | jun | mon | 8 | 999 | 0 | nonexisten | 1.4 | 94.465 | -41.8 | 4.961 | 5228.1 | no |
| 31 | technician | single | university.c | no | yes | no | cellular | may | tue | 1 | 999 | 0 | nonexisten | -1.8 | 92.893 | -46.2 | 1.266 | 5099.1 | yes |
| 41 | blue-collar | married | basic.9y | unknown | no | no | telephone | jun | fri | 3 | 999 | 0 | nonexisten | 1.4 | 94.465 | -41.8 | 4.967 | 5228.1 | no |
| 23 | services | single | high.schoc | no | no | no | cellular | jul | tue | 3 | 999 | 0 | nonexisten | 1.4 | 93.918 | -42.7 | 4.961 | 5228.1 | no |
| 39 | services | married | high.schoc | no | yes | yes | cellular | nov | mon | 3 | 999 | 0 | nonexisten | -0.1 | 93.2 | -42 | 4.191 | 5195.8 | no |
| 37 | technician | single | high.schoc | unknown | yes | no | telephone | may | thu | 2 | 999 | 0 | nonexisten | 1.1 | 93.994 | -36.4 | 4.86 | 5191 | no |
| 42 | managem | married | high.schoc | no | no | no | cellular | nov | thu | 2 | 999 | 0 | nonexisten | -0.1 | 93.2 | -42 | 4.076 | 5195.8 | no |
| 36 | managem | divorced | high.schoc | no | no | no | telephone | may | mon | 7 | 999 | 0 | nonexisten | 1.1 | 93.994 | -36.4 | 4.857 | 5191 | no |
| 43 | blue-collar | married | basic.9y | no | yes | no | cellular | nov | tue | 1 | 999 | 0 | nonexisten | -0.1 | 93.2 | -42 | 4.153 | 5195.8 | no |
| 21 | blue-collar | single | basic.4y | no | no | no | telephone | jun | fri | 1 | 999 | 0 | nonexisten | -2.9 | 92.963 | -40.8 | 1.268 | 5076.2 | no |
| 36 | admin. | divorced | university.c | no | yes | no | cellular | jul | mon | 1 | 999 | 0 | nonexisten | 1.4 | 93.918 | -42.7 | 4.96 | 5228.1 | no |
| 63 | retired | married | high.schoc | no | no | no | cellular | oct | wed | 1 | 999 | 0 | nonexisten | -3.4 | 92.431 | -26.9 | 0.74 | 5017.5 | yes |
| 72 | retired | married | basic.4y | no | no | yes | cellular | jul | fri | 4 | 8 | 1 | success | -1.7 | 94.215 | -40.3 | 0.822 | 4991.6 | yes |
| 52 | blue-collar | married | basic.9y | no | yes | no | cellular | may | fri | 3 | 999 | 0 | nonexisten | -1.8 | 92.893 | -46.2 | 1.25 | 5099.1 | no |
| 50 | blue-collar | married | basic.4y | unknown | no | no | telephone | jun | thu | 1 | 999 | 0 | nonexisten | 1.4 | 94.465 | -41.8 | 4.866 | 5228.1 | no |
| 38 | blue-collar | married | profession | no | no | no | telephone | may | mon | 1 | 999 | 0 | nonexisten | 1.1 | 93.994 | -36.4 | 4.857 | 5191 | no |
| 31 | admin. | single | university.c | no | no | no | cellular | jul | mon | 1 | 999 | 0 | nonexisten | 1.4 | 93.918 | -42.7 | 4.962 | 5228.1 | no |
| 28 | admin. | single | university.c | no | yes | no | cellular | oct | mon | 1 | 6 | 2 | success | -1.1 | 94.601 | -49.5 | 1.032 | 4963.6 | yes |
| 41 | blue-collar | married | high.schoc | no | yes | no | cellular | nov | tue | 1 | 999 | 0 | nonexisten | -0.1 | 93.2 | -42 | 4.153 | 5195.8 | no |
| 28 | student | single | high.schoc | unknown | no | no | telephone | jun | tue | 9 | 999 | 0 | nonexisten | 1.4 | 94.465 | -41.8 | 4.961 | 5228.1 | no |
| 42 | blue-collar | married | basic.6y | unknown | yes | no | telephone | may | mon | 3 | 999 | 0 | nonexisten | 1.1 | 93.994 | -36.4 | 4.857 | 5191 | no |

**图36：最终的数据集**

# 8 模型的评价

## 8.1 模型的优点

1. 在数据预处理过程中发现了数据不平衡的现象，并且采用了 SMOTE 过采样方法对数据进行重采样，大大提高了模型预测的精度。
2. 因为数据集中多是定类变量，采用了onehot编码处理，在一定程度上使模型的训练变得可行。
3. 在模型的求解过程中，使用了Adaboost、Logistic回归、支持向量机、决策树、随机森林等九个模型进行建模，然后通过比较这九种预测算法模型的分类准确率和AUC值，这样做加大了预测方法选择的客观性；后来又对这九种算法参数进行调整以及进行五折交叉验证，找到了分类准确率最高的模型参数，提高了模型的精确性，使结果更加符合实际意义。
4. 通过比较29种常见的分类算法模型，找出我们最佳的分类算法，选择有理有据。
5. 经过对比发现逻辑回归在准确率和时间上都比较优秀，因此选择LightGBM作为我们的最终预测模型，预测准确率达到了94%，且花费的时间较少，说明模型比较优秀！

## 8.2 模型的缺点

1. 对于数据不平衡问题，没有综合考虑欠采样和添加权重惩罚因子的方法。并没有明确的证据说明过采样是效果最好的方法。
2. 保留了所有特征。日后可以在特征选择上进行一个数据预处理的优化。

# 参考文献

[1] 徐树芳. 数值线性代数 [M]. 北京: 高等教育出版社, 2013.

[2] MESTOY P R, DUFF I S, KOSTER J, et al. A fully asynchronous multifrontal solver using distributed dynamic scheduling[J]. SIAM Journal on Matrix Analysis and Applications, 2001, 23(1);

[3] 李航.统计学习方法[M].北京:清华大学出版社,2012.