某移动通讯公司客户流失预警分析

一、背景介绍

目前在我国移动通讯行业基本呈现三足鼎立的局势，市场份额由中国移动、中国联通和中国电信三家运营商瓜分。

然而，随着电信企业之间的竞争加剧，电信运营商不断推出新的服务模式和业务，希望争取到更多的市场份额。但同时也在很大程度上加大了客户的不稳定性，使客户离网现象频繁发生。图 1 展示了三大通讯公司 2020 年 1 月到 6 月的客户月净增长数，从数据可以看出在 1 月和 2 月，中国移动和中国联通的客户流失较多，而 5 月中国电信苦户增长变为负数。因此，在客户流失方面，三大运营商都存在相关问题。

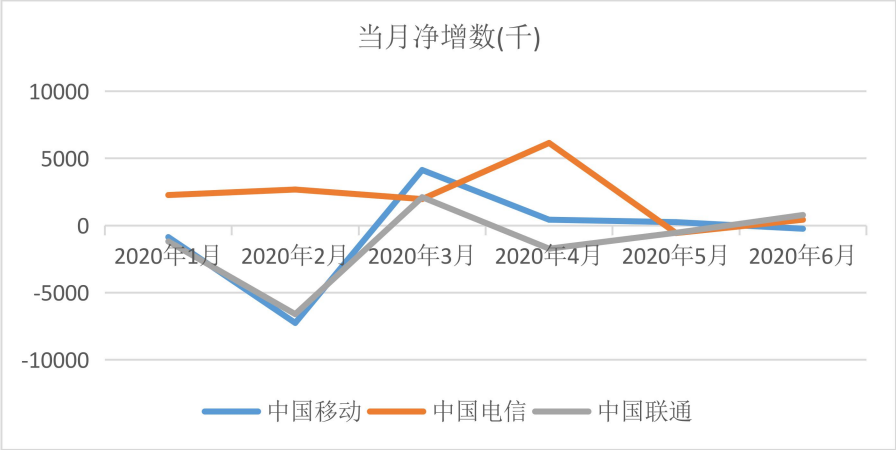

图 1 三大运营商 2020 年 1 月-6 月的当月净增客户数

研究表明，一个公司如果将其顾客流失率降低 5%，利润就能增加 25%~85%，可见大量的客户流失让运营商蒙受巨大损失。因此，如何防止客户流失、做好客户挽留工作，已成为运营商关注的焦点之一。

分析客户流失的原因，我们可以总结出以下几点：首先导致客户流失的外部原因可能有竞争对手的促销活动、客户趋重心理；导致客户流失的内部原因可能包括网络质量、服务质量、品牌等原因。因此针对这些可能的原因，面对有限的市场容量和不断上升的客户流失率，提出有效的减少客户流失的对策尤为重要。

鉴于实际意义和数据的可获得性，本文选取数据和指标进行建模与预测。我们试图进行数据分析、建立逻辑回归模型、模型预测的方式来探究某移动通讯公司客户流失问题，并提出商业建议。

二、数据来源和说明

本文的样本数据来自国内某运营商，数据已经进行了清理。数据提供了月度的基础通讯数据和通话详单数据，一共有包含 2 个，分别用于建模与预测，其中建模数据包含 48393 个观测，预测数据包含 47900 个观测，其中每条观测包含客户编号、在网时长、当月花费、个体的度、联系强度、个体信息熵、花费的变化、个体度的变化、是否流失 9 个变量。其中，我们将是否流失作为因变量，1 代表流失、0 代表非流失。又将在网时长、当月花费、个体的度、联系强度、个体信息熵、花费的变化、个体度的变化 7 个变量作为自变量，具体变量说明见表 1。

表 1 变量说明

	变量名称	变量类型	详细说明	备注
因变量（下月）	是否流失	分类变量	1=流失 0=非流失	流失率 1%
自变量（当月）	在网时长	连续变量	单位：天	数据截取日减入网时间
	当月花费	连续变量	单位：元	统计当月的总花费
	个体的度	连续变量	单位：人数	$D_i = \sum_{j=1} \alpha_{ij}$
	联系强度	连续变量	分钟/人	$T_i = \frac{Total\_Comm_i}{D_I}$
	个体信息熵	连续变量		$E_i = -\sum_{\alpha_{i,j}=1} p_{ij} * log(p_{ij})$
	个体度的变化	连续变量	单位：%	（当月个体的度-上月个体的度）/上月个体的度
	花费的变化	连续变量	单位%	（当月花费-上月花费）/上月花费

其中，我们将客户在这个网络中的社交资本定义为个体的度，$\alpha_{ij}=1$ 表示个体 i 和个体 j 通过电话，否则为 0。作为个体的度的补充，我们又定义了另外两个衍生变量，联系强度和个体信息熵。联系强度为个体与其他人之间的平均通话时长，用 T_i 来表示，个体信息熵则为通话分钟的分布，用 E_i 来表示，它的值越小，说明通话分布越集中。

三、描述性分析

将数据读入后，我们对数据进行描述性分析，考察了流失组和非流失组在各个指标上差异。利用 R 语言进行编程，得到结果。首先，我们对整体自变量数据进行描述性分析，得到每个自变量的均值、中位数、最小值、最大值，对数据进行初步的了解，如表 2:

表 2 整体自变量数据描述性分析

变量	均值	中位数	最小值	最大值
在网时长（天）	1257.4	994.7	184.0	4479.6
当月花费（元）	161.079	135.927	-2.764	511.055
个体的度（人）	66.6072	54.0370	-0.3713	303.9827
联系强度(分钟/人）	9.846	8.021	-2.355	62.509
个体信息熵	2.928	2.988	0.000	5.346
个体度的变化（%）	0.03693	0.00000	-0.99644	7.33333
花费的变化（%）	0.007203	0.000000	-1.000000	2.665025

（一）因变量：是否流失

我们关注因变量是否流失的分布情况，得到饼状图：

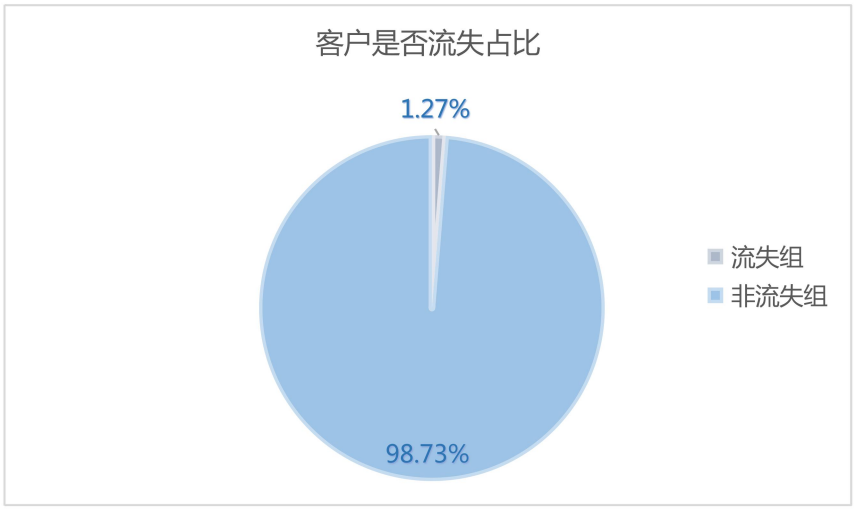

图 2 是否流失的饼状图

如图 2 所示，流失组的占比远远小于非流失组，因此现在的绝大部分的客户还是更愿意依赖运营商的。下面我们对一些因变量指标进行分析。

（二）自变量：在网时长和当月花费

我们对两个组别的在网时长和当月话费进行分析，这两个变量可以共同反应客户的活跃程度。利用箱线图进行分析，得到结果图如下：

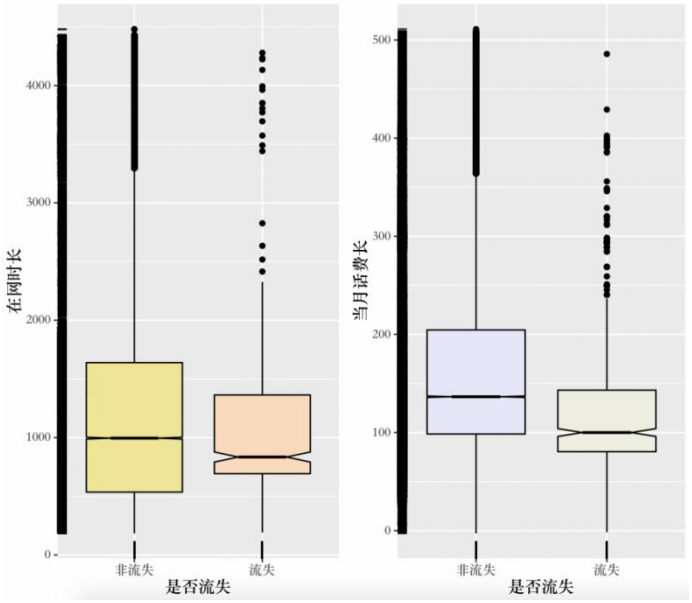

图 3 在网时长和当月花费的分组箱线图

由图 3 可以看出非流失组的在网时长和当月花费都要高于流失组的在网时长，这说明未流失客户更愿意在此运营商运营的网络下花更多的时间，并且花费的钱也更多，这与事实是相符的。

（二）自变量：个体的度

我们对个体的度进行分析，我们继续用箱线图进行分析，得到结果图如下：

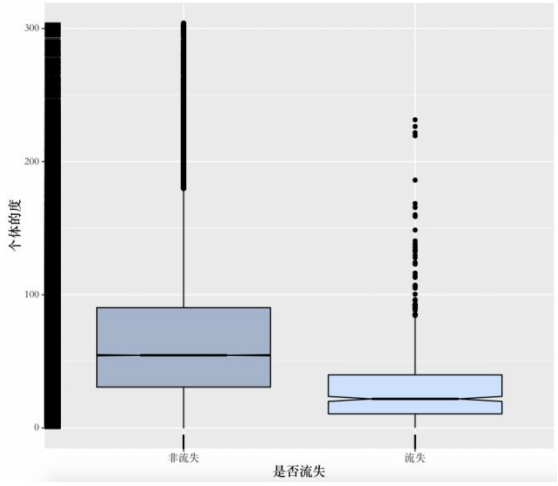

图 4 个体的度的分组箱线图

由图 4 可以看出非流失客户个体的度要明显高于流失组。这意味着非流失组使用该运营商进行通话的频率要更高，对该运营商的依赖性更强，因此不会流失。

（三）自变量：联系强度和个体信息熵

联系强度和个体信息熵可以很好的反映客户的社交集中情况。我们通过联系强度反映平均通话时长，通过个体信息熵反映通话分钟的分布。信息熵越大，说明通话时长的离散程度越大，说明客户跟不同的人打电话时长都比较平均。

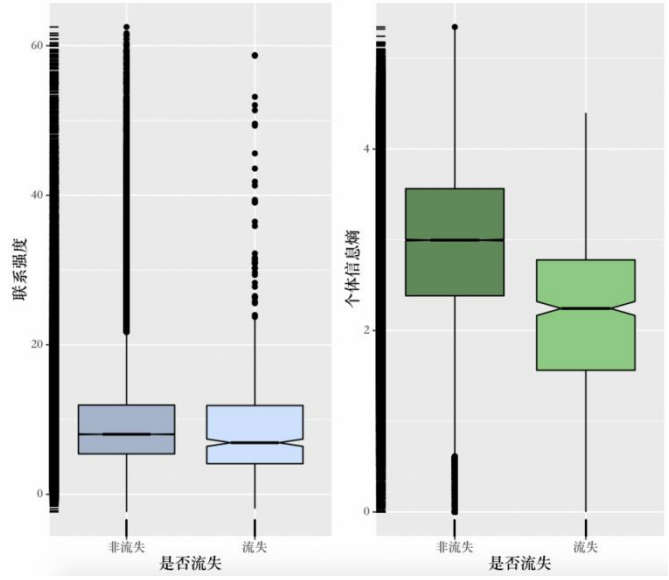

图 5 联系强度和个体信息熵的分组箱线图

由图 5 可以得到结果：非流失组的个体信息熵和通话时长都要多一些，这可以更好的解释非流失组使用该运营商的次数更多，通话离散程度更大跟不同的人打电话时长都比较平均，符合现实情况。

（四）自变量：花费和个体度的变化

我们对花费的变化和个体度的变化进行通话的动态分析。如图 4。

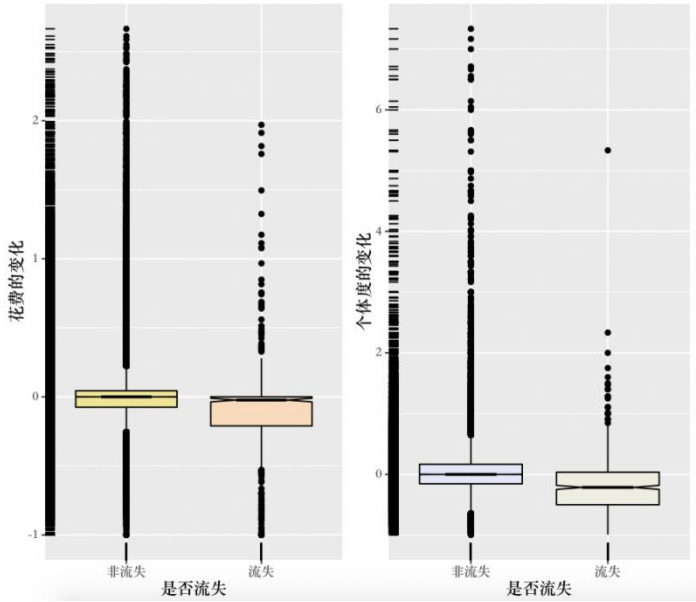

图 6 花费变化和个体度变化的分组箱线图

由图 6 可以得到，非流失组的两个指标变化都在 0 附近，而流失组的花费变

化略小于 0，个体度变化则明显为负值，这说明非流失组的稳定性更好一些，而流失组的离网率更多，在通话上花费的也少，与人群逐渐减少联系，这个结果与事实也相贴切。

综上所述，未流失客户要比流失客户通话更稳定，有着更多的联系人以及通话时间，在运营商的服务上也花费了更多的时间和金钱。未流失客户各方面频率均较为稳定。相对而言，流失客户的社交集中度较高，表明流失客户和运营商已经到了“冷战期”，客户正在逐渐减少需要联系的人，拨打更少的电话，这也解释了为什么流失客户显得并不活跃。

四、数据建模

（一）Logistic 回归模型

为了进一步开发客户流失预警模型帮助企业提前识别高风险流失客户，我们建立逻辑回归模型，其中因变量为 0-1 分类变量是否流失，自变量包括标准化后的在网时长、当月花费、个体的度、联系强度、个体信息熵、花费的变化、个体度的变化的自变量，回归模型的结果如下表：

表 3 回归结果（因变量：总阅读量）

变量名称	标准化系数估计	标准差	统计量	P 值
截距项	-5.043384288	0.07161676	-70.42184396	0 ***
在网时长	-0.226046176	0.059312722	-3.811090904	0.000138355***
当月花费	-0.29248944	0.058804974	-4.973889415	6.56E-07***
个体的度	-0.741340046	0.130229719	-5.692556583	1.25E-08***
联系强度	-0.223442869	0.041599232	-5.371321934	7.82E-08***
个体信息熵	-0.349399063	0.070394183	-4.96346496	6.92E-07***
花费的变化	-0.196579816	0.048329816	-4.06746457	4.75E-05***
个体度的变化	-0.377406515	0.051496149	-7.328829869	2.32E-13***

注：0 ‘***’

从回归结果可以看出，所有变量都对是否流失有显著影响，我们又建立了 AIC 和 BIC 模型，发现结果一致，说明 7 个变量都非常的重要。其 ROC 的面积为 0.775，如图 7，说明模型较好。

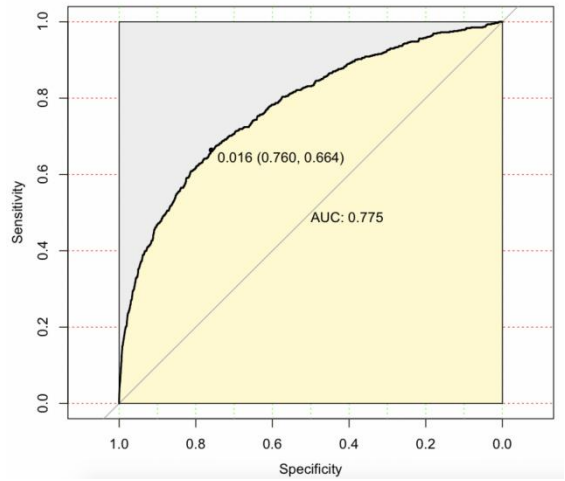

图 7 逻辑回归模型的 ROC 曲线

下面，我们对 Logistic 回归模型的结果进行解读。我们发现，在控制其他变量不变的情况下，7 个自变量的标准化回归系数均为负数，这说明随着自变量客户流失呈负相关，也就意味着随着自变量的增多，客户流失的概率就会减少。例如，在网时长的标准化系数为-0.226046176，因此当在网时长增加一天时，客户流失的对数发生比就减小 0.226046176。

因此我们可以总结出以下结果：在网时长越长，流失概率越低；当月花费越高，流失概率越低；个体的度越大，说明通话人数越多，流失概率越低；联系强度越大，说明平均通过人数越多，流失概率越低；个体信息熵越大，说明通话分布的越均匀，流失概率越低；个体度的变化变大，说明通话人数有所增加，流失概率变低；花费的变化变大，说明花费有所增加，流失概率变低。

通过对模型结果的解读可以发现，回归结果和描述性分析的结果相对一致。因此我们可以说模型的效果较好。

（二）模型预测与评估

为了进一步判断模型的正确性，我们接下来用测试集对模型进行预测，对比模型预测与真实的结果，计算混淆矩阵、TPR 和 FPR、绘制 ROC 和覆盖率-捕获率曲线，对模型进行评判。

1） ROC 曲线、AUG 值与最佳阈值

对于逻辑回归模型，我们需要去寻找一个最佳阈值来将流失组和与非流失组进行更准确的划分，ROC 曲线可以很好的利用 TPR 和 FPR 的平衡关系对最佳阈值进行选择，这里 TPR 指的是在所有实际为阳性的样本中，被正确地判断为阳性之比率；FPR 指的是在所有实际为阴性的样本中，被错误地判断为阳性之比率。因此我们可以很容易的得到当 TPR 高而 FPR 低时，证明这个模型是比较好的。同时我们还可以通过 ROC 曲线的面积 AUG 值来判断模型的拟合程度。下面，我们对测试集的 ROC 曲线进行绘制，得到结果如图 7：

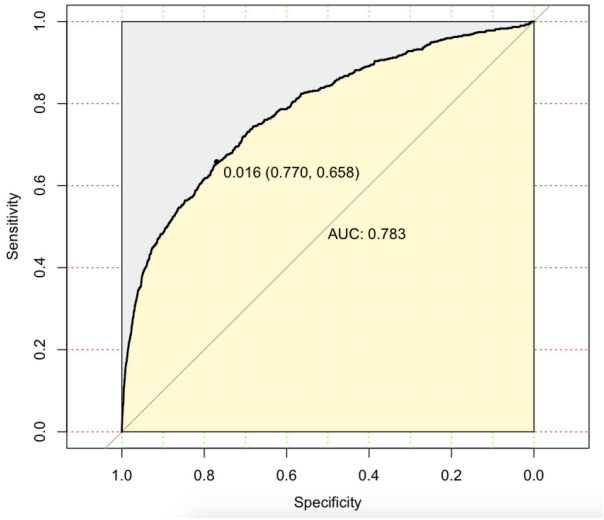

图 8 模型预测的 ROC 曲线

图 8 表明在模型预测中，ROC 曲线的弧度靠近左上角，模型的 AUG 值为 0.783，模型的拟合程度较好。预测的最佳阈值为 0.016，这说明当预测概率大于 0.016 的时候，预测为流失，当预测概率小于 0.016 的时候，预测为非流失。

2） 混淆矩阵

对于逻辑回归模型，混淆矩阵用于比较分类结果和实际测得值，可以很好的反映分类结果的精度。不同的阈值也会产生不同的混淆矩阵，我们已经得到最佳阈值为 0.016，因此将最佳阈值带入模型可以得到混淆矩阵及相关结果，如表 4:

表 4 最佳阈值的混淆矩阵

		预测值		
		非流失	流失	总计
真实值	非流失	36089	237	36326
	流失	11118	456	11574
	总计	47207	693	47900

由表 4 可以看出整体的错判率为(237+11118)/47900=23.7%，TPR 为 456/11574=3.9%，FPR 为 237/36326=0.65%。

3） 准确率

我们已经进行了最佳阈值选择，为了更直观的看出模型的好坏，我们对模型准确率进行计算，这是对模型好坏评估最直接的一种方法，最终得到结果为 76.3%，说明模型的预测能力较强，可以认为模型性能良好。

4） 覆盖率-捕获率曲线

在实际中，我们还经常用成本收益曲线来度量的成本和收益。该曲线关乎两个度量成本和收益的指标——覆盖率和捕获率。覆盖率是指预测为流失组人数与总人数的比值，我们把它当成一个衡量成本的指标；而捕获率则是指预测为流失组人数中实际的确是流失组人数和总人数的比值，可以作为衡量收益的指标。我们希望成本越低、收益越高越好。因此，捕获率和覆盖率的关系跟 ROC 曲线中的 TPR 和 FPR 的关系很类似。下图为模型的覆盖率-捕获率曲线，我们发现其走势与 ROC 曲线类似，也反映了较好的结果。

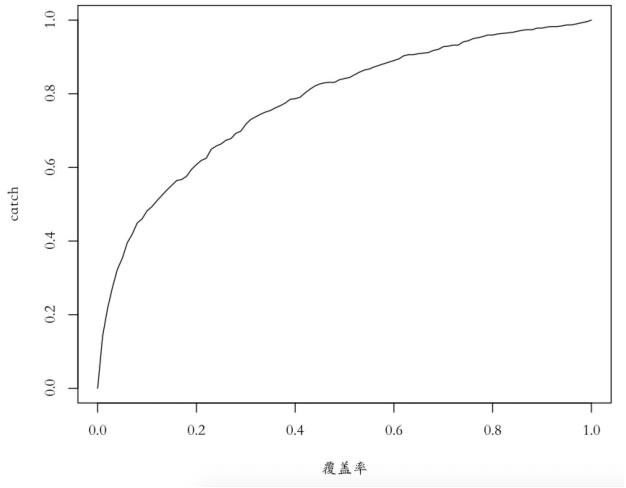

图 9 覆盖率-捕获率曲线

其中，在覆盖率为 20%的时候，捕获率可以达到 60%，说明覆盖预测概率最高的前 20%人，可以抓住 60%的流失客户，说明模型精度较高。

五、结论与模型探索

通过上述结果，我们可以为运营商开发一个客户流失预警模型，从而帮助企业更好的进行客户关系管理，对高风险客户做好客户关怀，尽最大努力挽留，可以加强企业抗客户流失风险的能力。在未来，企业可以设立一套基于该模型的流失预警体系，根据成本预算来选择不同的覆盖率，可以对客户进行实时的打分预测，一旦预测的流失概率超过了设定的阈值，预警体系就会发出警告，告诉企业该重点关注该客户。

此外，我们关注 Logistic 回归模型的性能体现虽然较好，但是当观察混淆矩阵时，在流失组的预测上仍然展现出了不完美。因此，模型的结果整体理想，但是仍有可以改进的地方。为了进一步对模型进行探索，我们采用了几种可能提高模型的方法进行了更深一步的研究。

1、特征离散化

Logistic 回归模型对于连续性的数值特征的输入，通常需要对特征做归一化输出为在 0-1 之间的数，这样可以加速模型计算及训练收敛，在本文中，我们也用了这样的方法，可以看出归一化对模型的建立起到了一定的积极作用。但是，在工业界，研究人员很少直接将连续值作为逻辑回归模型的特征输入，而是先将连续特征离散化，然后做（Onehot、WOE）编码再输入模型。这样做的原因是逻辑回归是广义线性模型，模型对特征线性的加权求和，通过 sigmoid 归一化为概率。这样的特征表达是很有限的。对于非线性关系，它就无法表达的很好。因此，通过对特征进行离散化，可以增加模型的非线性表达，提高了拟合能力。

常用的特征离散化方法有等宽、等频、卡方分箱、决策树分箱等方式，分箱的差异也直接影响着模型效果，在此我们选择 Kmeans 分箱法将 7 个自变量数据进行分箱，利用 Python 对自变量分布进行可视化，我们分别将在网时长、个体的度、联系强度、个体信息熵、花费的变化、个体度的变化划分为 2 类，将当月花费划分为 3 类。接着利用上述四中方法进行建模、预测，得到 ROC 曲线和混淆矩阵分别如图 11 和表 5:

表 5 特征离散化后的最佳阈值混淆矩阵

		预测值		
		非流失	流失	总计
真实值	非流失	27419	160	27579
	流失	19788	533	20321
	总计	47207	693	47900

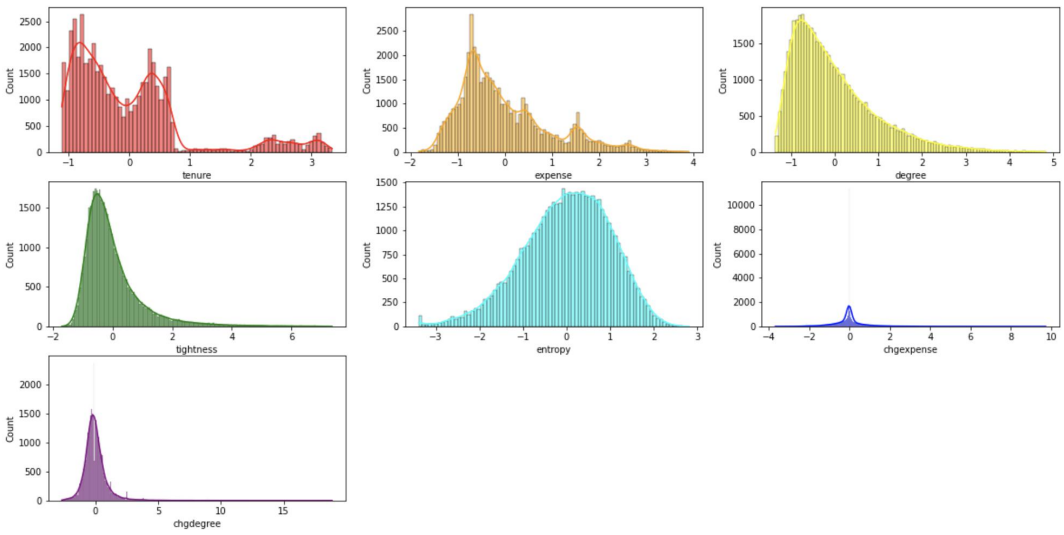

图 10 自变量分布图

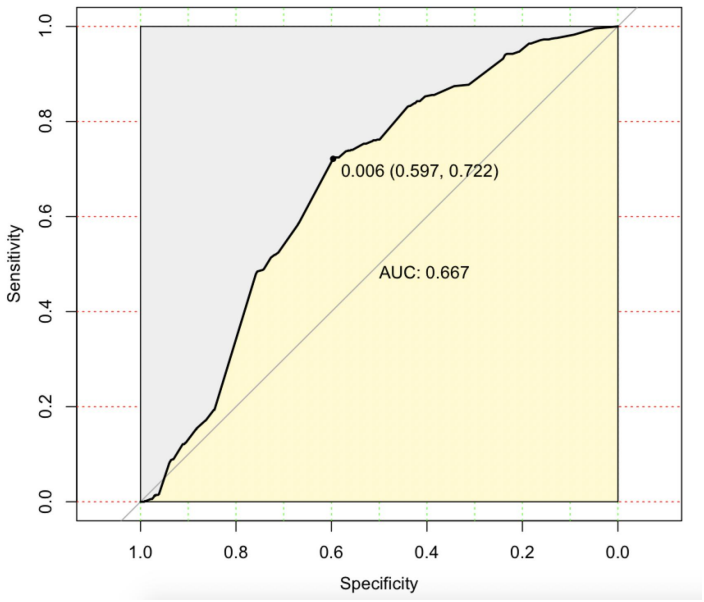

图 11 特征离散化后模型的 ROC 曲线

由以上的结论均可以看出模型并没有原来的好，因此我们还是选择最开始的模型。

2、AdaBoost 算法

AdaBoost 是一种迭代算法，其核心思想是针对同一个训练集训练不同的分类器(弱分类器)，然后把这些弱分类器集合起来，构成一个更强的最终分类器（强分类器）。它可以有效的提高模型的精度，还可以弱化学习效果，防止过拟合。我们考虑在 AdaBoost 的框架下，进行 Logistic 回归模型的多次学习，完成预测模型构建。通过 R 语言，我们设定迭代次数为 10 次，得到 AUG 值仅有 0.515，说明模型远远没有原来的好，因此我们还是选择最开始的模型。

综上所述，我们对模型进行了探索，但是发现最开始建立的模型都显示出更

好的结果，因此，我们认为原先的模型预测效果可以令人满意，可以将该模型进一步应用于流失预警中。

附录 1　R 语言程序代码

```
##读入数据
options (warn = -1)
dt <- read.csv('/Users/yuqinhan1229/Desktop/sampledata.csv')
head(dt)
summary(dt)
###导入包
library(ggplot2)
library(gridExtra)
library(DAAG)
library(pROC)
###用 ggplot 画 churn 与 ID 的饼状图
n= 48393
m = sum(dt["churn"])
B <- c("流失组", "非流失组")
A = c(m,n-m)
myLabel = as.vector(B)
myLabel = paste(myLabel,"(", round(A / n, 2), "%)", sep = "")
pie(A,labels=myLabel,main = "simple pie chart",family='Hiragino Sans GB W3')
###对分类变量 churn 进行命名
attach(dt)
dt$churn[churn==0]="非流失"
dt$churn[churn==1]="流失"
detach(dt)



p = ggplot(dt, aes(x = "", y = A, fill = myLabel)) + #创建坐标轴
   geom_bar(stat = "identity") +
   coord_polar(theta = "y")     +
   labs(x = "", y = "", title = "") +
   theme(text=element_text(family="Songti SC",size=8,face = "bold"))
print(p) #显示饼图

###用 ggplot 画 churn 与 tnure 的箱线图
a1= ggplot(dt, aes(x=churn, y=tenure)) + xlab("是否流失")+ylab("在网时长")+
   geom_boxplot(fill=c('khaki','peachpuff'),color="black", notch=TRUE)+
```

```
   geom_rug(color="black")+
   theme(text=element_text(family="Songti SC",size=12,face = "bold"))
###用 ggplot 画 churn 与 expense 的箱线图
a2=ggplot(dt, aes(x=churn, y=expense)) + xlab("是否流失")+ylab("当月话费长")+
   geom_boxplot(fill=c('lavender','ivory2'),
                color="black", notch=TRUE)+
   geom_rug(color="black")+
   theme(text=element_text(family="Songti SC",size=12,face = "bold"))
grid.arrange(a1,a2,ncol=2) ##并列输出

###用 ggplot 画 churn 与 degree 的箱线图
ggplot(dt, aes(x=churn, y=degree)) + xlab("是否流失")+ylab("个体的度")+
   geom_boxplot(fill="aliceblue",
                color="black", notch=TRUE)+
   geom_rug(color="black")+
   theme(text=element_text(family="Songti SC",size=12,face = "bold"))
###用 ggplot 画 churn 与 tightness 的箱线图
a3=ggplot(dt, aes(x=churn, y=tightness)) +xlab("是否流失")+ylab("联系强度")+
   geom_boxplot(fill=c("lightsteelblue3","lightsteelblue1"),
                color="black", notch=TRUE)+
   geom_rug(color="black")+
   theme(text=element_text(family="Songti SC",size=12,face = "bold"))
###用 ggplot 画 churn 与 entropy 的箱线图
a4= ggplot(dt, aes(x=churn, y=entropy)) +xlab("是否流失")+ylab("个体信息熵")+
   geom_boxplot(fill=c("palegreen4","palegreen3"),
                color="black", notch=TRUE)+
   geom_rug(color="black")+
   theme(text=element_text(family="Songti SC",size=12,face = "bold"))
grid.arrange(a3,a4,ncol=2)

###用 ggplot 画 churn 与 chgexpense 的箱线图
a5= ggplot(dt, aes(x=churn, y=chgexpense)) +xlab("是否流失")+ylab("花费的变化
")+
   geom_boxplot(fill=c('khaki','peachpuff'),
                color="black", notch=TRUE)+
   geom_rug(color="black")+
   theme(text=element_text(family="Songti SC",size=12,face = "bold"))
```

```
###用 ggplot 画 churn 与 chgdegree 的频数图
a6= ggplot(dt, aes(x=churn, y=chgdegree)) +xlab("是否流失")+ylab("个体度的变化
")+
   geom_boxplot(fill=c('lavender','ivory2'),
                        color="black", notch=TRUE)+
   geom_rug(color="black")+
   theme(text=element_text(family="Songti SC",size=12,face = "bold"))
grid.arrange(a5,a6,ncol=2)
a5

###重新读入数据拟合 logis 模型
dt <- read.csv('/Users/yuqinhan1229/Desktop/sampledata.csv')
dt[,2:8] = scale(dt[,2:8]) ##模型进行标准化
fm<-glm(churn~tenure+expense+degree+tightness+entropy+chgexpense+chgdegree,f
amily=binomial(link="logit"),data=dt)
summary(fm)

AIC(fm)   #AIC
BIC(fm)   #BIC
fm.AIC=step(fm,trace=0)
summary(fm.AIC)   #经过 AIC 选择后的变量系数输出表

n=length(dt[,1])
fm.BIC<-step(fm,scope = list(lower = fm, upper = fm), direction = "both", trace =
TRUE, k = log(n))
summary(fm.BIC)   #经过 BIC 选择后的变量系数输出表

x1=predict(fm,newdata=dt,type="response")   # 计算 y=1 的概率的预测值
modelroc1=roc(dt$churn,x1)
x2=predict(fm.AIC,newdata=dt,type="response")   # 计算 y=1 的概率的预测值
modelroc2=roc(dt$churn,x2)
x3=predict(fm.BIC,newdata=dt,type="response")   # 计算 y=1 的概率的预测值
modelroc3=roc(dt$churn,x3)
##绘制 roc 曲线
plot(modelroc1,print.auc=TRUE,auc.polygon=TRUE,grid=c(0.1,0.2),grid.col=c("gree
n","red"),max.auc.polygon=TRUE,auc.polygon.col="lemonchiffon",print.thres=TRU
E)
```

```
lines(modelroc2,col="red")
lines(modelroc3, col = "blue")


###对模型进行预测
dtpre <- read.csv('/Users/yuqinhan1229/Desktop/preddata.csv')##读入预测数据
dtpre[,2:8] = scale(dtpre[,2:8]) ##模型进行标准化
p=predict(fm,newdata=dtpre,type="response")  # 计算 y=1 的概率的预测值
p1 = 1*(p>mean(dtpre$churn)) ##设置阈值
table(p1,dtpre$churn)
modelroc=roc(dtpre$churn,p)
##绘制 roc 曲线
plot(modelroc,print.auc=TRUE,auc.polygon=TRUE,grid=c(0.1,0.2),grid.col=c("green
","red"),max.auc.polygon=TRUE,auc.polygon.col="lemonchiffon",print.thres=TRUE
)
p2 = 1*(p>0.016) ##设置最佳阈值
table(p2,dtpre$churn)

###计算 TPR,FPR
data=data.frame(prob=p,obs=dtpre$churn)
data=data[order(data$prob),]
n=nrow(data)
tpr=fpr=rep(0,n)
for (i in 1:n){
  threshold=data$prob[i]
  tp=sum(data$prob>threshold&data$obs==1)
  fp=sum(data$prob>threshold&data$obs==0)
  tn=sum(data$prob)
  fn=sum(data$prob)
  tpr[i]=tp/(tp+fn)  #真正率
  fpr[i]=fp/(tn+fp)  #假正率

}
plot(fpr,tpr,type='l')

###评估模型的预测效果
p.results <- ifelse( p> 0.012,"1","0")
```

```
misClasificError <- mean(p.results != dtpre$churn)
print(paste('Accuracy',1-misClasificError))

##绘制覆盖率-捕获率曲线
list = seq(0,1,0.1)
tol = sum(dtpre$churn)
catch = sapply(list, function(x){
   x1=quantile(p,1-x)
   res = sum(dtpre$churn[p>x1])/tol
   return(res)
})
par(family='STKaiti')
plot(list,catch,type="l",xlab="覆盖率",ylabel="捕获率")


##更深一步探索
##将个体度变化和花费变化离散化
dt <- read.csv('/Users/yuqinhan1229/Desktop/sampledata.csv')
attach(dt)
dt$chgexpense[chgexpense<=0]= 0
dt$chgexpense[chgexpense>0]= 1
dt$chgdegree[chgdegree<=0]= 0
dt$chgdegree[chgdegree>0]= 1
detach(dt)
dt[,2:6] = scale(dt[,2:6]) ##模型进行标准化
fm<-glm(churn~tenure+expense+degree+tightness+entropy+chgexpense+chgdegree,f
amily=binomial(link="logit"),data=dt)
summary(fm)
x1=predict(fm,newdata=dt,type="response")   # 计算 y=1 的概率的预测值
modelroc1=roc(dt$churn,x1)
plot(modelroc1,print.auc=TRUE,auc.polygon=TRUE,grid=c(0.1,0.2),grid.col=c("gree
n","red"),max.auc.polygon=TRUE,auc.polygon.col="lemonchiffon",print.thres=TRU
E)
###对模型进行预测
dtpre <- read.csv('/Users/yuqinhan1229/Desktop/preddata.csv')##读入预测数据
attach(dtpre)
dtpre$chgexpense[chgexpense<=0]= 0
```

```
dtpre$chgexpense[chgexpense>0]= 1
dtpre$chgdegree[chgdegree<=0]= 0
dtpre$chgdegree[chgdegree>0]= 1
detach(dtpre)
dtpre[,2:6] = scale(dtpre[,2:6]) ##模型进行标准化
p=predict(fm,newdata=dtpre,type="response")   # 计算 y=1 的概率的预测值
p1 = 1*(p>mean(dtpre$churn)) ##设置阈值
table(p1,dtpre$churn)
modelroc=roc(dtpre$churn,p)
##绘制 roc 曲线
plot(modelroc,print.auc=TRUE,auc.polygon=TRUE,grid=c(0.1,0.2),grid.col=c("green
","red"),max.auc.polygon=TRUE,auc.polygon.col="lemonchiffon",print.thres=TRUE
)
p2 = 1*(p>0.018) ##设置最佳阈值
table(p2,dtpre$churn)



##特征离散化
library(discretization)
dt <- read.csv('/Users/yuqinhan1229/Desktop/sampledata.csv')
dtpre <- read.csv('/Users/yuqinhan1229/Desktop/preddata.csv')##读入预测数据
dt[,2:8] = scale(dt[,2:8]) ##模型进行标准化
dtpre[,2:8] = scale(dtpre[,2:8]) ##模型进行标准化
# 离散化计算
# 导入无监督分箱包——infotheo
library(infotheo)
# 分成几个区域
nbins <- 2
# kmeans 分箱法，先给定中心数，将观察点利用欧式距离计算与中心点的距离进
行归类，再重新计算中心点，直到中心点# 不再发生变化，以归类的结果做为分
箱的结果。
# 将 chgexpense 数据分 3 份，并以 1、2、3 赋值
k_means <- kmeans(dt$tenure , 3)
# 对分箱进行赋值
dt$tenure= k_means$cluster
# 将 tightnesse 数据分 2 份，并以 1、2 赋值
```

```
k_means <- kmeans(dt$tightness , nbins)
# 对分箱进行赋值
dt$tightness= k_means$cluster
# 将 expense 数据分 3 份，并以 1、2、3 赋值
k_means <- kmeans(dt$expense, 3)
# 对分箱进行赋值
dt$expense= k_means$cluster
# 将 degree 数据分 2 份，并以 1、2 赋值
k_means <- kmeans(dt$degree, nbins)
# 查看各分类数量
table(k_means$cluster)
# 对分箱进行赋值
dt$degree= k_means$cluster
# 将 entropy 数据分 2 份，并以 1、2 赋值
k_means <- kmeans(dt$entropy, nbins)
# 对分箱进行赋值
dt$entropy= k_means$cluster
# 将 chgexpense 数据分 2 份，并以 1、2 赋值
k_means <- kmeans(dt$chgexpense, nbins)
# 查看各分类数量
table(k_means$cluster)
# 对分箱进行赋值
dt$chgexpense= k_means$cluster
# 将 chgdegree 数据分 2 份，并以 1、2 赋值
k_means <- kmeans(dt$chgdegree, nbins)
# 对分箱进行赋值
dt$chgdegree= k_means$cluster

###重新读入数据拟合 logis 模型
fm<-glm(churn~tenure+expense+degree+tightness+entropy+chgexpense+chgdegree,f
amily=binomial(link="logit"),data=dt)
summary(fm)

x1=predict(fm,newdata=dt,type="response")  # 计算 y=1 的概率的预测值
modelroc1=roc(dt$churn,x1)
##绘制 roc 曲线
plot(modelroc1,print.auc=TRUE,auc.polygon=TRUE,grid=c(0.1,0.2),grid.col=c("gree
```

```
n","red"),max.auc.polygon=TRUE,auc.polygon.col="lemonchiffon",print.thres=TRU
E)


###对模型进行预测
# 分成几个区域
nbins <- 2
# kmeans 分箱法，先给定中心数，将观察点利用欧式距离计算与中心点的距离进行归类，再重新计算中心点，直到中心点# 不再发生变化，以归类的结果做为分箱的结果。
# 对分箱进行赋值
dtpre$tenure= k_means$cluster
# 将 tightnesse 数据分 2 份，并以 1、2 赋值
k_means <- kmeans(dtpre$tightness , nbins)
# 对分箱进行赋值
dtpre$tightness= k_means$cluster
# 将 expense 数据分 3 份，并以 1、2、3 赋值
k_means <- kmeans(dtpre$expense, 3)
# 对分箱进行赋值
dtpre$expense= k_means$cluster
# 将 degree 数据分 2 份，并以 1、2 赋值
k_means <- kmeans(dtpre$degree, nbins)
# 查看各分类数量
table(k_means$cluster)
# 对分箱进行赋值
dtpre$degree= k_means$cluster
# 将 entropy 数据分 2 份，并以 1、2 赋值
k_means <- kmeans(dtpre$entropy, nbins)
# 对分箱进行赋值
dtpre$entropy= k_means$cluster
# 将 chgexpense 数据分 2 份，并以 1、2 赋值
k_means <- kmeans(dtpre$chgexpense, nbins)
# 查看各分类数量
table(k_means$cluster)
# 对分箱进行赋值
dtpre$chgexpense= k_means$cluster
# 将 chgdegree 数据分 2 份，并以 1、2 赋值
```

```
k_means <- kmeans(dtpre$chgdegree, nbins)
# 对分箱进行赋值
dtpre$chgdegree= k_means$cluster
p=predict(fm,newdata=dtpre,type="response")  # 计算 y=1 的概率的预测值
modelroc=roc(dtpre$churn,p)
##绘制 roc 曲线
plot(modelroc,print.auc=TRUE,auc.polygon=TRUE,grid=c(0.1,0.2),grid.col=c("green
","red"),max.auc.polygon=TRUE,auc.polygon.col="lemonchiffon",print.thres=TRUE
)
p2 = 1*(p>0.006) ##设置最佳阈值
table(p2,dtpre$churn)


###ada boost 算法
####导入包
library(adabag)
library(openxlsx)
library(adabag)
library(pROC) #绘制 ROC 曲线
dt <- read.csv('/Users/yuqinhan1229/Desktop/sampledata.csv')
dtpre<- read.csv('/Users/yuqinhan1229/Desktop/preddata.csv')
#将 2 个数据集分别命名为训练集和测试集
train_data = dt[2:9]
test_data = dtpre[2:9]
train_data[1:8]=scale(train_data[1:8])
test_data[1:8]=scale(test_data[1:8])
#数据预处理
train_data$churn = as.factor(train_data$churn)
test_data$churn = as.factor(test_data$churn)
#Adaboost 算法
dt_adaboost <-
boosting(churn~tenure+expense+degree+tightness+entropy+chgexpense+chgdegree,
                                    data = train_data,boos=TRUE, mfinal=50 )
##预测测试集
pre_decisiontree_ada <- predict(dt_adaboost,newdata = test_data)$class
#将测试集计算所得概率与观测本身取值整合到一起
obs_p_decision_ada= data.frame(prob=pre_decisiontree_ada,obs=test_data$churn)
```

```
#输出混淆矩阵
table(test_data$churn,pre_decisiontree_ada,dnn=c("真实值","预测值"))
#绘制 ROC 图像
decisiontree_roc_ada <- roc(test_data$churn,as.numeric(pre_decisiontree_ada))
plot(decisiontree_roc_ada, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1,
0.2),grid.col=c("green", "red"), max.auc.polygon=TRUE,auc.polygon.col="skyblue",
print.thres=TRUE,main='Adaboost 算法 ROC 曲线')
p2 = 1*(pre_decisiontree_ada>0.018) ##设置最佳阈值
table(p2,dtpre$churn)
```

附录 2　Python 程序代码

```
#查看自变量分布图
import pandas as pd
import numpy as np
import random
import math
from sklearn.model_selection import train_test_split
data = pd.read_csv('/Users/yuqinhan1229/Desktop/sampledata.csv')
dt=data.iloc[:,1:8]
dt.head()
dt = (dt - np.mean(dt)) / np.std(dt)
dt['churn']= data['churn']
## Distribution of continuous variables
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
## Find the continuous variable and set the color.
figure = plt.figure(figsize=(20, 10))
continues =
["tenure","expense","degree","tightness","entropy","chgexpense","chgdegree"]
colors = ['red', 'orange', 'yellow', 'green', 'cyan', 'blue', 'purple']
for i in range(1, 8):
    plt.subplot(3,3,i)
    sns.histplot(dt[continues[i-1]], color = colors[i-1], kde=True)
    plt.xlabel(continues[i-1])
```

```
plt.plot()
```