

## 金融行业交叉销售之应用浅析

华院分析技术（上海）有限公司 模型工程师 姜建勋

[摘要] 随着金融行业的逐步放开和竞争的加剧，金融行业的营销管理越来越重视采用新的营销方法和工具以提升市场营销能力。交叉销售在这其中正在发挥越来越重要的作用，本文将结合具体案例阐述如何在金融行业中应用数据挖掘的工具发现交叉销售的机会。

关键词：交叉销售，数据挖掘，频繁集，预测模型，神经网络，算法，决策树。

### Multistage Cross-Sell Model of Clients in the Financial Industry

#### ABSTRACT

This paper details the steps to develop a multistage cross-sell model of clients in the Financial Services industry. This model can be used to score the likelihood of a customer to purchase multiple products. Topics covered include data preparation for datamining, several advanced modeling techniques, and multistage cross-sell model score comparison. Strong non-linear curvatures among the input variables were found during the modeling process and examined. The modeling techniques used are Decision Tree, Neural Network, Regression in SAS Enterprise Miner (version 4.1) of Windows 2000 system.

#### Keywords:

Cross-sell, Datamining, Frequent Item Set, Predictive Modeling, ANN, Algorithm, Decision Tree

- 背景:

在金融服务行业，对客户进行有效地交叉销售和客户保持正变得日益重要。近年来数据挖掘技术在各行业都取得了令人振奋地应用，尤其是在金融服务行业，数据挖掘技术的成功推广使得交叉销售变得更加富有成效和可操作。通过对多种预测模型的设计和应用，可以非常好地来识别客户对多种金融产品的购买倾向。为了成功地开发模型，通常情况下需要以下的数据来源：客户的交易数据——主要包含客户购买的频率和数量以及购买顺序；客户的自然属性——主要包含年龄、性别、收入、教育背景、职业状况等。

在实际的应用阶段，通过数据挖掘模型得到的结果将被用来指明客户购买多种金融产品的可能性以及可能的购买顺序。这样通过一系列的业务逻辑将其应用就能达到在合适的时间向合适的人群推荐合适的产品之目标，最终获得的将是更高的客户反馈和营销成功率，随之而来的还有更高的客户满意度。

在本文中，将对以下产品进行分析：汽车贷款，全国通，卡消费，基金，保险，外汇宝，储蓄业务。主要的分析平台是 SAS Enterprise Miner，

其中涉及到的主要的数据挖掘算法主要有决策树、逻辑斯蒂回归、神经网络、关联规则。开发过程遵循 SAS 的 SEMMA (Sample, Explore, Modification, Modeling, Assessment) 方法论, 在此过程中通过多种学习机的逼近可以发现复杂的非线性关系, 同时利用模型评估模块可以很好地确定学习机的容量和复杂度, 以期在推广能力和准确性之间取得良好的平衡。

- 建模目标:  
本文涉及到的建模过程有以下主要目标:
  1. 预测客户购买多种产品的可能性
  2. 确定客户购买多种金融产品的序列
  3. 给客户评分以确定购买可能性之高低
- 数据源:  
产品信息—产品层次、产品定义、产品描述等  
客户信息——性别、年龄、职业、行业、教育背景、收入、开户日期、已购买产品等  
交易信息—购买时间、购买数量、购买频率; 消费、投资、储蓄、转账、理财等
- 建模:  
本阶段建模的主要目的是发现哪些客户能成为多种金融产品的潜在使用者, 并给出可能性高低的评分。
  1. 为了保证模型的充分代表性, 充足的数据采样就十分重要。同时为了使模型能更好地发现合适的规则, 本模型的开发采用了分层抽样的策略, 按照目标变量的先验概率从全体中构建建模数据集。该建模数据集被随机分层的分成三个部分: 训练集、验证集、测试集。各自所占的比例如下: 训练集 50%、验证集 30%、测试集 20%。由于也应用了分层抽样, 因此三个数据集中目标变量的先验概率相同。
  2. 为了获得最佳的性能, 以上数据集分别由系统自动的生成面向挖掘的专有数据集。
  3. 在数据中通常会有很多缺失值、空值、错误值等, 如果继续保留这些记录将会对最终模型的准确性和推广性产生不良影响。通常针对缺失值有如下方法可以使用: 删除包含缺失值的记录、用平均值或众数进行替换、用回归法进行替换等。本模型开发过程中采用的使用平均值进行替换。具体实现的时候采用 SAS Enterprise Miner 的 Replacement 节点进行替换。决策树结点中采用 'tree imputation with surrogates' 模式。这样当一个变量的缺失值过多时, 决策树将会自动采用预测能力近似的其他具有较少缺失值得变量以作为代替。同时生成树的时候会同时产生替代的分裂规则。这种替换是递归的, 当一个变量的缺失值过多时, 系统自动采用下一个替代的变量生成分裂规则, 而当本变量也有很多缺失值时, 系统将继

寻找下一个替代变量。同时，决策树本身具有良好的适应性，从技术上讲，决策树可以接受缺失值，如果缺失值的确具有强的预测力，那么决策树将为缺失值自动生成一个分支作为一种规则。

4. 在本阶段建模过程中将用到决策树 **Decision Tree**，神经网络 **Neural Network**，逻辑斯蒂回归 **Logistic Regression** 三种学习机来预测哪些客户将成为多种金融产品的潜在购买者。三要素构成一种学习机：学习任务；性能指标（或者目标函数）；训练经验。

在此处的学习目标定义如下：购买了多种金融产品的客户定义为“1”，没有购买多种金融产品的客户定义为“0”。训练经验主要指学习经验的来源，此处的来源正如前文讲到的数据来源，即基于如下假设：客户的购买倾向可以通过客户的交易行为来体现，同时过去的行为模式在未来类似情况下具有较大的可能性重演。所以我们建模的出发点就是通过研究过去购买多种产品的客户的行为模式来预测客户未来的购买倾向。性能指标可以因学习机之不同而异，通常对于回归模型常用的目标函数是最小均方差，对于决策树通常采用的目标函数是基尼指标，熵减。另外对于回归树会用到最小均方差作为目标函数。

具体来讲，决策树提供了一种展示类似在什么条件下会得到什么值这类规则的方法。比如，在贷款申请中，要对申请的风险大小做出判断，图 1 是为了解决这个问题而建立的一棵决策树，从中我们可以看到决策树的基本组成部分：决策节点、分支和叶子。

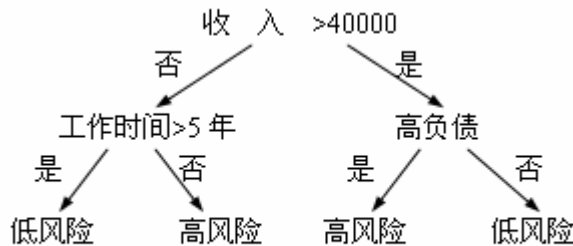


图 1：一棵简单的决策树

决策树中最上面的节点称为根节点，是整个决策树的开始。

本例中根节点是“收入 > ¥40,000”，对此问题的不同回答产生了“是”和“否”两个分支。

决策树的每个节点子节点的个数与决策树在用的算法有关。如 CART 算法得到的决策树每个节点有两个分支，这种树称为二叉树。允许节点含有多于两个子节点的树称为多叉树。

每个分支要么是一个新的决策节点，要么是树的结尾，称为叶子。在沿着决策树从上到下遍历的过程中，在每个节点都会遇到一个问题，对每个节点上问题的不同回答导致不同的分支，最后会到达一个叶子节点。这个过程就是利用决策树进行分类的过程，利用几个变量（每个变量对应一个问

题) 来判断所属的类别(最后每个叶子会对应一个类别)。

假如负责借贷的银行官员利用上面这棵决策树来决定支持哪些贷款和拒绝哪些贷款,那么他就可以用贷款申请表来运行这棵决策树,用决策树来判断风险的大小。

“年收入 $>$ ¥40,000”和“高负债”的用户被认为是“高风险”,同时“收入 $<$ ¥40,000”但“工作时间 $>$ 5年”的申请,则被认为“低风险”而建议贷款给他/她。

数据挖掘中决策树是一种经常要用到的技术,可以用于分析数据,同样也可以用来作预测(就像上面的银行官员用他来预测贷款风险)。常用的算法有CHAID、CART、Quest和C5.0。

建立决策树的过程,即树的生长过程是不断的把数据进行切分的过程,每次切分对应一个问题,也对应着一个节点。对每个切分都要求分成的组之间的“差异”最大。

各种决策树算法之间的主要区别就是对这个“差异”衡量方式的区别。对具体衡量方式算法的讨论超出了本文的范围,在此我们只需要把切分看成是把一组数据分成几份,份与份之间尽量不同,而同一份内的数据尽量相同。这个切分的过程也可称为数据的“纯化”。看我们的例子,包含两个类别——低风险和高风险。如果经过一次切分后得到的分组,每个分组中的数据都属于同一个类别,显然达到这样效果的切分方法就是我们所追求的。

到现在为止我们所讨论的例子都是非常简单的,树也容易理解,当然实际中应用的决策树可能非常复杂。假定我们利用历史数据建立了一个包含几百个属性、输出的类有十几种的决策树,这样的一棵树对人来说可能太复杂了,但每一条从根结点到叶子节点的路径所描述的含义仍然是可以理解的。决策树的这种易理解性对数据挖掘的使用者来说是一个显著的优点。然而决策树的这种明确性可能带来误导。比如,决策树每个节点对应分割的定义都是非常明确毫不含糊的,但在实际生活中这种明确可能带来麻烦(凭什么说年收入¥40,001的人具有较小的信用风险而¥40,000的人就没有)。

建立一颗决策树可能只要对数据库进行几遍扫描之后就能完成,这也意味着需要的计算资源较少,而且可以很容易的处理包含很多预测变量的情况,因此决策树模型可以建立得很快,并适合应用到大量的数据上。

对最终要拿给人看的决策树来说,在建立过程中让其生长的太“枝繁叶茂”是没有必要的,这样既降低了树的可理解性和可用性,同时也使决策树本身对历史数据的依赖性增大,也就是说这是这棵决策树对此历史数据可能非常准确,一旦应用到新的数据时准确性却急剧下降,我们称这种情况为训练过度。为了使得到的决策树所蕴含的规则具有普遍意义,必须防

止训练过度，同时也减少了训练的时间。因此我们需要有一种方法能让我们在适当的时候停止树的生长。常用的方法是设定决策树的最大高度（层数）来限制树的生长。还有一种方法是设定每个节点必须包含的最少记录数，当节点中记录的个数小于这个数值时就停止分割。

与设置停止增长条件相对应的是在树建立好之后对其进行修剪。先允许树尽量生长，然后再把树修剪到较小的尺寸，当然在修剪的同时要求尽量保持决策树的准确度尽量不要下降太多。

对决策树常见的批评是说其在为一个节点选择怎样进行分割时使用“贪心”算法。此种算法在决定当前这个分割时根本不考虑此次选择会对将来的分割造成什么样的影响。换句话说，所有的分割都是顺序完成的，一个节点完成分割之后不可能以后还有机会回过头来再考察此次分割的合理性，每次分割都是依赖于他前面的分割方法，也就是说决策树中所有的分割都受根结点的第一次分割的影响，只要第一次分割有一点点不同，那么由此得到的整个决策树就会完全不同。那么是否在选择一个节点的分割的同时向后考虑两层甚至更多的方法，会具有更好的结果呢？目前我们知道的还不是很清楚，但至少这种方法使建立决策树的计算量成倍的增长，因此现在还没有哪个产品使用这种方法。

而且，通常的分割算法在决定怎么在一个节点进行分割时，都只考察一个预测变量，即节点用于分割的问题只与一个变量有关。这样生成的决策树在有些本应很明确的情况下可能变得复杂而且意义含混，为此目前新提出的一些算法开始在一个节点同时用多个变量来决定分割的方法。比如以前的决策树中可能只能出现类似“收入 $< ¥35,000$ ”的判断，现在则可以用“收入 $< (0.35 * \text{抵押})$ ”或“收入 $> ¥35,000$  或抵押 $< 150,000$ ”这样的问题。

决策树很擅长处理非数值型数据，这与神经网络只能处理数值型数据比起来，就免去了很多数据预处理工作。甚至有些决策树算法专为处理非数值型数据而设计，因此当采用此种方法建立决策树同时又要处理数值型数据时，反而要做把数值型数据映射到非数值型数据的预处理。

神经网络近来越来越受到人们的关注，因为它为解决大复杂度问题提供了一种相对来说比较有效的简单方法。神经网络可以很容易的解决具有上百个参数的问题（当然实际生物体中存在的神经网络要比我们这里所说的程序模拟的神经网络要复杂的多）。神经网络常用于两类问题：分类和回归。

在结构上，可以把一个神经网络划分为输入层、输出层和隐含层（见图4）。输入层的每个节点对应一个个的预测变量。输出层的节点对应目标变量，可有多个。在输入层和输出层之间是隐含层（对神经网络使用者来说不可见），隐含层的层数和每层节点的个数决定了神经网络的复杂度。

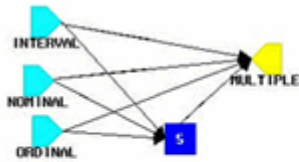


图 2：SAS 中的神经网络结构

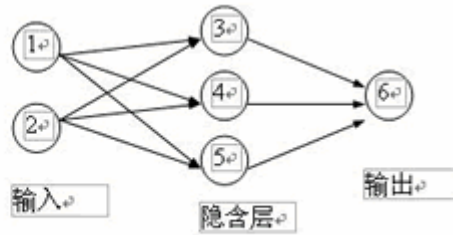


图 3：一个神经网络

除了输入层的节点，神经网络的每个节点都与很多它前面的节点（称为此节点的输入节点）连接在一起，每个连接对应一个权重  $W_{xy}$ ，此节点的值就是通过它所有输入节点的值与对应连接权重乘积的和作为一个函数的输入而得到，我们把这个函数称为活动函数或挤压函数。如图 5 中节点 4 输出到节点 6 的值可通过如下计算得到：

$$W_{14} * \text{节点 1 的值} + W_{24} * \text{节点 2 的值}$$

神经网络的每个节点都可表示成预测变量（节点 1, 2）的值或值的组合（节点 3-6）。注意节点 6 的值已经不再是节点 1、2 的线性组合，因为数据在隐含层中传递时使用了活动函数。实际上如果没有活动函数的话，神经元网络就等价于一个线性回归函数，如果此活动函数是某种特定的非线性函数，那神经网络又等价于逻辑回归。

调整节点间连接的权重就是在建立（也称训练）神经网络时要做的工作。最早的也是最基本的权重调整方法是错误回馈法，现在较新的有变化坡度法、类牛顿法、Levenberg-Marquardt 法、和遗传算法等。无论采用那种训练方法，都需要有一些参数来控制训练的过程，如防止训练过度和控制训练的速度。

决定神经网络拓扑结构（或体系结构）的是隐含层及其所含节点的个数，以及节点之间的连接方式。要从头开始设计一个神经网络，必须要决定隐含层和节点的数目，活动函数的形式，以及对权重做那些限制等，当然如果采用成熟软件工具的话，他会帮你决定这些事情。

在诸多类型的神经网络中，最常用的是前向传播式神经网络，也就是我们前面图示中所描绘的那种。我们下面详细讨论一下，为讨论方便假定只含有一层隐含节点。

可以认为错误回馈式训练法是变化坡度法的简化，其过程如下：

前向传播：数据从输入到输出的过程是一个从前向后的传播过程，后一节点的值通过它前面相连的节点传过来，然后把值按照各个连接权重的大小加权输入活动函数再得到新的值，进一步传播到下一个节点。

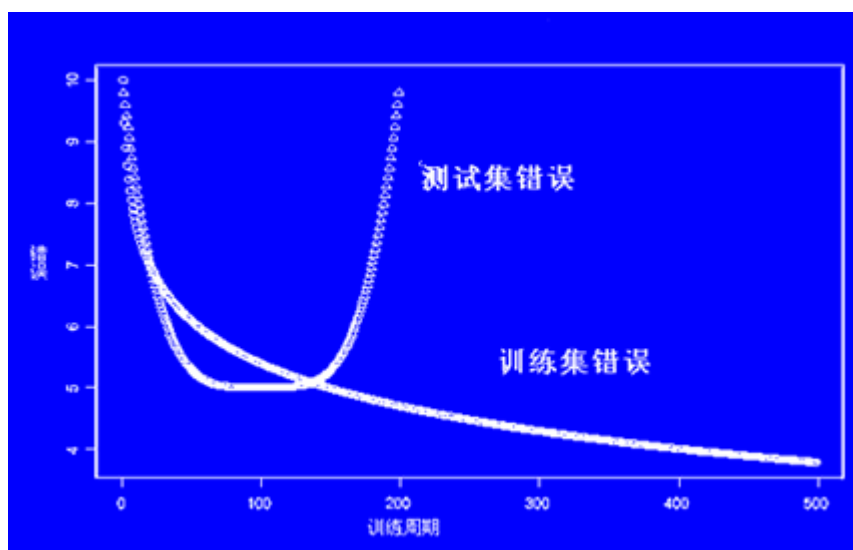
回馈：当节点的输出值与我们预期的值不同，也就是发生错误时，神经网络就要“学习”（从错误中学习）。我们可以把节点间连接的权重看成后一节点对前一节点的“信任”程度（他自己向下一节点的输出更容易

受他前面哪个节点输入的影响)。学习的方法是采用惩罚的方法,过程如下:如果一节点输出发生错误,那么他看他的错误是受哪个(些)输入节点的影响而造成的,是不是他最信任的节点(权重最高的节点)陷害了他(使他出错),如果是则要降低对他的信任值(降低权重),惩罚他们,同时升高那些做出正确建议节点的信任值。对那些收到惩罚的节点来说,他也需要用同样的方法来进一步惩罚它前面的节点。就这样把惩罚一步步向前传播直到输入节点为止。

对训练集中的每一条记录都要重复这个步骤,用前向传播得到输出值,如果发生错误,则用回馈法进行学习。当把训练集中的每一条记录都运行过一遍之后,我们称完成一个训练周期。要完成神经网络的训练可能需要很多个训练周期,经常是几百个。训练完成之后得到的神经网络就是在通过训练集发现的模型,描述了训练集中响应变量受预测变量影响的变化规律。

由于神经网络隐含层中的可变参数太多,如果训练时间足够长的话,神经网络很可能把训练集的所有细节信息都“记”下来,而不是建立一个忽略细节只具有规律性的模型,我们称这种情况为训练过度。显然这种“模型”对训练集会有很高的准确率,而一旦离开训练集应用到其他数据,很可能准确度急剧下降。为了防止这种训练过度的情况,我们必须知道在什么时候要停止训练。在有些软件实现中会在训练的同时用一个测试集来计算神经网络在此测试集上的正确率,一旦这个正确率不再升高甚至开始下降时,那么就认为现在神经网络已经达到做好的状态了可以停止训练。

图 6 中的曲线可以帮我们理解为什么利用测试集能防止训练过度的出现。在图中可以看到训练集和测试集的错误率在一开始都随着训练周期的增加不断降低,而测试集的错误率在达到一个谷底后反而开始上升,我们认为这个开始上升的时刻就是应该停止训练的时刻。



神经网络和统计方法在本质上有很多差别。神经网络的参数可以比统计方法多很多。如图 4 中就有 13 个参数(9 个权重和 4 个限制条件)。由

于参数如此之多，参数通过各种各样的组合方式来影响输出结果，以至于很难对一个神经网络表示的模型做出直观的解释。实际上神经网络也正是当作“黑盒”来用的，不用去管“盒子”里面是什么，只管用就行了。在大部分情况下，这种限制条件是可以接受的。比如银行可能需要一个笔迹识别软件，但他没必要知道为什么这些线条组合在一起就是一个人的签名，而另外一个相似的则不是。在很多复杂度很高的问题如化学试验、机器人、金融市场的模拟、和语言图像的认识，等领域神经网络都取得了很好的效果。神经网络的另一个优点是很容易在并行计算机上实现，可以把他的节点分配到不同的 CPU 上并行计算。

在使用神经网络时有几点需要注意：

第一，神经网络很难解释，目前还没有能对神经网络做出显而易见的解释的方法学。

第二，神经网络会学习过度，在训练神经网络时一定要恰当的使用一些能严格衡量神经网络的方法，如前面提到的测试集方法和交叉验证法等。这主要是由于神经网络太灵活、可变参数太多，如果给足够的时间，他几乎可以“记住”任何事情。

第三，除非问题非常简单，训练一个神经网络可能需要相当可观的时间才能完成。当然，一旦神经网络建立好了，在它做预测时运行时还是很快得。

第四，建立神经网络需要做的数据准备工作量很大。一个很有误导性的神话就是不管用什么数据神经网络都能很好的工作并做出准确的预测。这是不确切的，要想得到准确度高的模型必须认真的进行数据清洗、整理、转换、选择等工作，对任何数据挖掘技术都是这样，神经网络尤其注重这一点。比如神经网络要求所有的输入变量都必须是 0-1（或-1 -- +1）之间的实数，因此像“地区”之类文本数据必须先做必要的处理之后才能用作神经网络的输入。

回归是通过具有已知值的变量来预测其他变量的值。在最简单的情况下，回归采用的是象线性回归这样的标准统计技术。但在大多数现实世界中的问题是不能用简单的线性回归所能预测的。如商品的销售量、股票价格、产品合格率等，很难找到简单有效的方法来预测，因为要描述这些事件的变化所需的变量以上百计，且这些变量本身往往都是非线性的。为此人们又发明了许多新的手段来试图解决这个问题，如逻辑斯蒂回归，对于二分因变量 Y 的编码一般采用 0, 1 编码，1 表示 yes，0 表示 no，用 p 表示 1 在整个数据集中出现的概率即  $p=P(Y=1)$ ；同时  $1-p=P(Y=0)$ 。问题是我们如何对 p 建立模型呢？

首先我们的策略可能是拟合一个如下的简单线性回归模型。

$$p = a + b_1 X_1$$

但是会出现下面一些问题：Y 的预测值可能会出现负数、大于 1 等情况。这与实际的 p 值是不符合的。我们可能用下面的模型来解决上述的问题。

$$p = e^{a + b_1 X_1}$$

但是它也不是一个非常好的策略。尽管它不会产生一个负的 p 值估计，但是它依然会导致大于 1 的 P 值估计。最后我们采取下面的模型就可以完全避免前面出现的问题。

$$p = \frac{e^{a+b_1 X_1}}{1 + e^{a+b_1 X_1}}$$

经过适当的变化后

$$\ln \left[ \frac{p}{1-p} \right] = a + b_1 X_1$$

其图形如下，这样概率就被挤压在 [0, 1] 之间，因此 sigmoid 函数也被称为挤压函数。

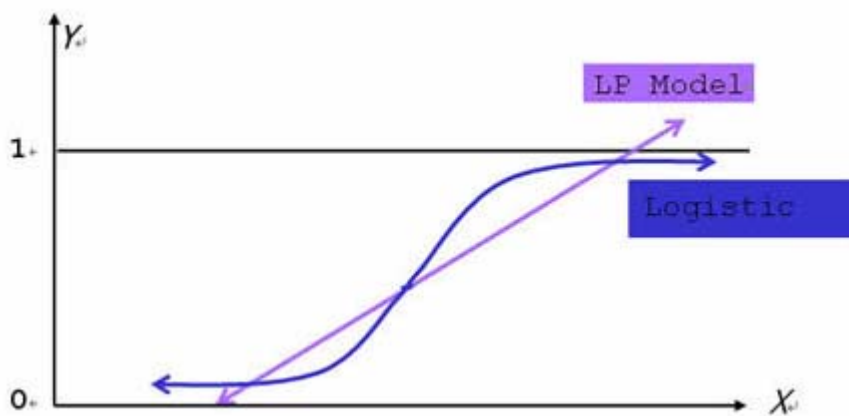


图 5：逻辑斯蒂回归与线性回归比较

5. 在模型开发完成后，经过 SAS Assessment 模块的评估，发现在本阶段建模中决策树具有最高的提升度，具体讲就是将后验概率按照降序排列，在前 30% 发现决策树的预测力最强，也就是能发现最多的潜在客户。在本例中共有 9 个决策树结点具有较高的预测力。下面的代码用来计算每一个节点的提升度指标。

```
procsql;
create table scored as select a._node_ , a.single, b.multiple
from (select _node_, count(dunsno) as single
from &_score
where multiple='Single'
```

```

group by _node_ ) a
left join
(select _node_ , count(dunsno) as multiple
from &_score
where multiple='Multiple'
group by _node_ ) b
on a._node_ = b._node_ ;
quit;
data scored;
set scored;
if multiple=. then multiple=0;
total=single+multiple;
score=int(multiple/total*10000)/100;
run;

```

下面是来自 Enterprise Miner Diagram 的工作流：

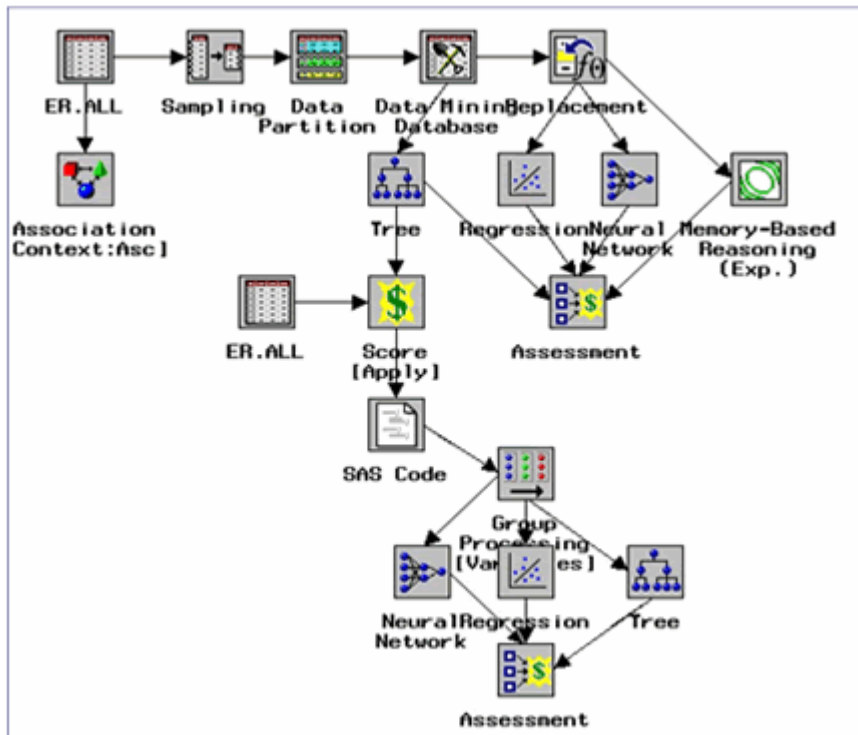


图 6：SAS Enterprise Miner 中的工作流

6. 下面的表格描述了决策树每一个节点的划分规则以及对应的提升度：

节点	提升度
开户时间 >30 个月 and ↓ 活期储蓄余额:[50000,150000] and ↓ 月均交易次数>20	1.99
半年内活期定期储蓄与额之比>.25 and ↓ 月均消费金额: (1500,4000] and ↓ 月均转账次数:[2,5]	1.62
外汇交易次数:[12,21] and ↓ 银证通交易次数:[14,30]	1.49
教育背景:[博士、硕士、本科] and ↓ 月均交易次数:(23,40) and ↓ 活期储蓄余额:[35,60]	1.15
开户时间<14 个月 and ↓ 活期储蓄余额:[3000,9000] ↓ 教育背景:[专科、职高]	0.95
月均交易次数:[15,28] ↓ 半年内活期定期储蓄与额之比:[3.5,7]	0.56
月均交易次数:(10,20) and ↓ 活期储蓄余额:[35,60]	0.36

表 1：决策数节点规则及其提升度指标

在上表中，前四个节点的提升度指标大于 1，这样这四个节点中的客户被用来参与下一阶段建模，也就是从中发现产品购买顺序以及产品购买之间的关联度。SAS Enterprise Miner 中的 Association 节点将完成此项任务。这在整个建模过程中可以算作第二阶段。从数据上看，产品购买日期也将参与建模用以表明购买顺序。经过运行之后发现最强的 15 个购买序列举例来说第一行表示在所有的交易中先购买产品 A 然后购买产品 B 的频率是 1109 ，同时后面的百分比表示在所有购买产品 A 的客户中有 5.71% 的客户购买了产品 B。

购买序列	频率	百分比
Product A ==> Product B	1,109	5.71%
Product B ==> Product A	874	2.13%
Product A ==> Product C	531	2.73%
Product C ==> Product A	422	1.84%
Product A ==> Product D	343	1.77%
Product A ==> Product E	313	1.61%
Product B & Product C ==> Product A	280	1.89%
Product A ==> Product F	279	1.44%
Product A ==> Product B & Product C	279	1.44%
Product F ==> Product A	244	1.21%
Product A ==> Product B & Product E	243	1.25%
Product A ==> Product B & Product D	243	1.25%
Product E ==> Product A	240	2.77%
Product D ==> Product A	238	3.68%
Product B ==> Product C	234	0.57%

表 2: 产品购买序列

7. 在本阶段建模过程中, 主要目的是估计已经购买了特定产品的客户再购买一种新产品的后验概率, 例如已经购买了产品 A 的客户再购买产品 B 的后验概率或者已经购买了产品 B 的客户再购买产品 A 的后验概率等等。正如上表所示, 每一个购买序列都对应一种预测性模型, 但不是每一种模型都有很强的预测力和稳定性, 因此综合考虑多种因素之后最终选择六个模型作为最后的结果加以应用。这样在最后的 application 中就呈如下形式: 每一个客户都有 7 个评分, **cross-sell score** 表示该客户购买多种产品的可能性高低, 分数越高表示越有可能同时购买多种产品。后面的六个分数分别表明了如果已经购买了特定产品紧接着购买另外一种产品的可能性, 如果有的分数为零表示该客户还没有购买序列的前导产品

Customer ID	Cross sell score	Product A --> Product B	Product B --> Product A	Product A --> Product C	Product C --> Product A	Product A --> Product D	Product A --> Product E
3201767	8.72	6.79	0	0.23	0	5.26	5.1
1905289	8.72	3.25	0	0.62	0	7.98	6.65
6706752	8.72	0	4.85	0.15	6.23	0.62	1.52
1008754	7.13	1.65	4.85	0.23	3.5	5.26	1.52
2709015	7.13	2.76	3.21	4.52	0	3.31	1.52
Etc.							

表 3: 客户购买倾向评分表

- 应用:  
当模型开发完成后, 市场人员可以利用上表提供的支持做出更好的决策。这样市场部门在制定营销策略的时候就能充分考虑客户的购买倾向已制定更有针对性的活动计划, 因此整个市场活动管理将变得十分定制化并且

重点突出。因此对于提升市场活动的效率将会很有帮助。

- **结论:**

本文通过一个具体案例比较粗浅的讲明了如何利用多种模型来解决交叉销售的应用。说明了整个过程中关键的两部分：发现购买多种产品的可能性；根据客户当前购买产品预测其购买其他特定产品的可能性。这种方法将营销活动从细分客户群的粒度下降到客户级，因此可以认为具有更加定制化和差异化的特性。这与通常的“一对一”营销已经十分接近了。作为一种新兴的营销管理方法，要取得好的效果一定需要实践的检验。同时从这种方法本身来讲需要比较多的 IT 系统支持和人员培训。只有在信息流被良好管理的情况下，以数据挖掘为基础的营销方法才能在实践中取得成功。

- **参考文献:**

Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, New York: Oxford University Press.

Breiman, L., Friedman, J. H., Olshen, R.A., and Stone, C. J. (1984), *Classification and Regression Trees*, Chapman and Hall.

Enterprise Miner Software, Online Tutorial (SAS V8).

Ripley, B.D. (1996), *Pattern Recognition and Neural Networks*, Cambridge University Press.

Rud, Olivia Parr (2001), *Data Mining Cookbook*, New York: John Wiley & Sons.

Zahavi, J. and Levin, N. (1997), "Applying Neural Computing to Target Marketing," *Journal of Direct Marketing*, 11, 5-22.

Tom M. Mitchell, *Machine Learning*, McGRAW-HILL International Editions

Two Crows Introduction to Data Mining and Knowledge Discovery Third Edition