信用卡评分

2016 年 1 月 10 日

一、数据准备

1、 问题的准备

- 目标：要完成一个评分卡，通过预测某人在未来两年内将会经历财务危机的可能性来提高信用评分的效果，帮助贷款人做出最好的决策。

- 背景：

– 银行在市场经济中起到至关重要的作用。他们决定谁在什么条件下可以得到融资，并且可以创造或打破投资决策。而市场、社会，以及个人和企业都需要获得贷款。

– 信用评分算法，对默认可能性进行猜测，这是银行用来判断贷款是否应该被授予的方法。

- 准备：

– 首先是基于个人借贷的场景，确定“违约”的定义： 根据新的 Basel II Capital Accord（巴塞尔二资本协议），一般逾期 90 天算作违约。

– 在判别指标上，选择使用历史最大违约天数。

2、数据的获取与整合

- 数据来源：数据来自 Kaggle，cs-training.csv 是有 15 万条的样本数据，下图可以看到这份数据的大致情况。下载地址为：
https://www.kaggle.com/c/GiveMeSomeCredit/data

- 数据描述：数据属于个人消费类贷款，只考虑评分卡最终实施时能够使用到的数据应从如下一些方面获取数据：

– 基本属性：包括了借款人当时的年龄。

– 偿债能力：包括了借款人的月收入、负债比率。

– 信用往来：两年内 35-59 天逾期次数、两年内 60-89 天逾期次数、两年内 90 天或高于 90 天逾期的次数。

– 财产状况：包括了开放式信贷和贷款数量、不动产贷款或额度数量。

– 贷款属性：暂无。

– 其他因素：包括了借款人的家属数量（不包括本人在内）。

• 原始变量：

变量名	变量类型	变量描述
SeriousDlqin2yrs	Y/N	超过 90 天或更糟的逾期拖欠
RevolvingUtilizationOf UnsecuredLines	percentage	无担保放款的循环利用：除了不动产和像车贷那样除以信用额度总和的无分期付款债务的信用卡和个人信用额度总额
age	integer	借款人当时的年龄
NumberOfTime30-59DaysPastDueNotWorse	integer	35-59 天逾期但不糟糕次数
DebtRatio	percentage	负债比率
MonthlyIncome	real	月收入
NumberOf OpenCreditLinesAndLoans	integer	开放式信贷和贷款数量，开放式贷款（分期付款如汽车贷款或抵押贷款）和信贷（如信用卡）的数量
NumberOfTimes90DaysLate	integer	90 天逾期次数：借款者有 90 天或更高逾期的次数
NumberRealEstateLoans OrLines	integer	不动产贷款或额度数量：抵押贷款和不动产放款包括房屋净值信贷额度
NumberOfTime60-89DaysPastDueNotWorse	integer	60-89 天逾期但不糟糕次数：借款人在在过去两年内有 60-89 天逾期还款但不糟糕的次数
NumberOfDependents	integer	家属数量：不包括本人在内的家属数量

• 时间窗口：自变量的观察窗口为过去两年，因变量表现窗口为未来两年。

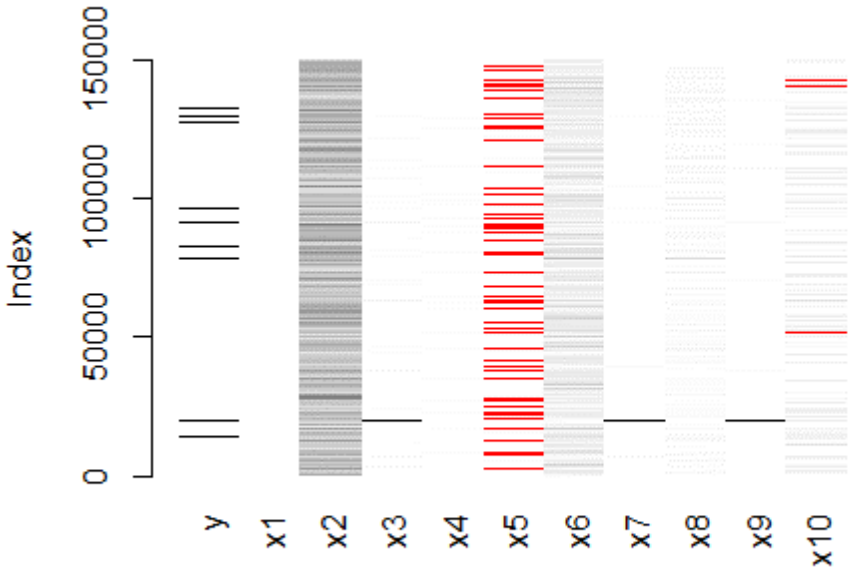
二、数据处理

首先去掉原数据中的顺序变量，即第一列的 id 变量。由于要预测的是 SeriousDlqin2yrs 变量，因此将其设为响应变量 y，其他分别设为 x1~x10 变量。

1、缺失值分析及处理

在得到数据集后，我们需要观察数据的分布情况，因为很多的模型对缺失值敏感，因此观察是否有缺失值是其中很重要的一个步骤。在正式分析前，我们先通过图形进行对观测字段的缺失情况有一个直观的感受。

```
matrixplot(traindata)
```




```
md.pattern(traindata)
```

```
##        y x1 x2 x3 x4 x6 x7 x8 x9  x10    x5
## 120269 1  1  1  1  1  1  1  1  1    1     1     0
##  25807 1  1  1  1  1  1  1  1  1    1     0     1
##   3924 1  1  1  1  1  1  1  1  1    0     0     2
##        0  0  0  0  0  0  0  0  0 3924 29731 33655
```

利用 matrixplot 函数对缺失值部分进行可视化展示，上图中浅色表示值小，深色表示值大，而默认缺失值为红色。因此可以看到 x5 变量和 x10 变量，即 MonthlyIncome 变量和 NumberOfDependents 两个变量存在缺失值，具体确实情况可以见上表，monthlyincome 列共有缺失值 29731 个，numberofdependents 有 3924 个。

对于缺失值的处理方法非常多，例如基于聚类的方法，基于回归的方法，基于均值的方法，其中最简单的方法是直接移除，但是在本文中因为缺失值所占比例较高，直接

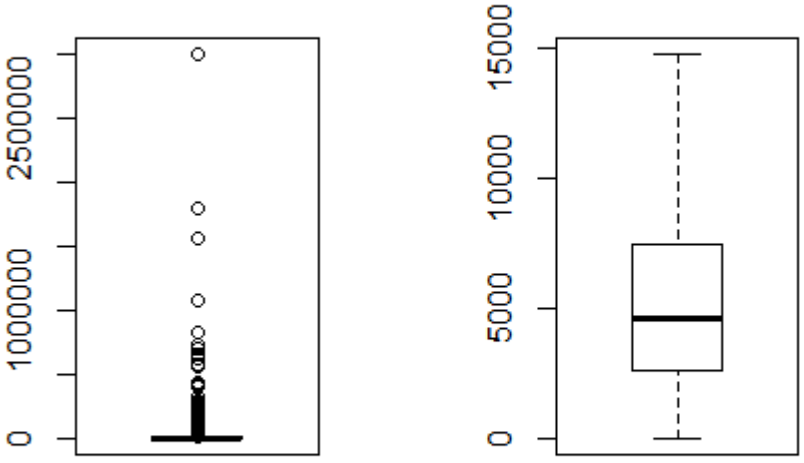
移除会损失大量观测，因此并不是最合适的方法。在这里，我们使用 KNN 方法对缺失值进行填补。

traindata<-knnImputation(traindata,k=10,meth = "weighAvg")

2、异常值分析及处理

关于异常值的检测，这里简单介绍以下一些检测方法：

- 单变量异常值检测：在 R 语言中使用函数 boxplot.stats()可以实现单变量检测，该函数根据返回的统计数据生成箱线图。在上述函数的返回结果中，有一个参数 out，它是由异常值组成的列表。更明确的说就是里面列出了箱线图中箱须线外面的数据点。比如我们可以查看月收入分布，第一幅图为没有删除异常值的箱线图。第二幅箱线图删除异常值后，可以发现月收入主要集中分布在 3000-8000 之间。但是在这份分析报告中，因为我们对业务尚不熟悉，不好将大于 8000 的数据直接归为异常值，因此对该变量未做处理。



- 使用 LOF（局部异常因子）检测异常值：LOF（局部异常因子）是一种基于密度识别异常值的算法。算法实现是：将一个点的局部密度与分布在它周围的点的密度相比较，如果前者明显的比后者小，那么这个点相对于周围的点来说就处于一个相对比较稀疏的区域，这就表明该点事一个异常值。LOF 算法的缺点是它只对数值型数据有效。包 ‘DMwR’ 和包 ‘dprep’ 中的 lofactor()可以计算 LOF 算法中的局部异常因子。

- 通过聚类检测异常值：检测异常值的另外一种方式就是聚类。先把数据聚成不同的类，选择不属于任何类的数据作为异常值。例如，基于密度的聚类 DBSCAN 算法的实现就是将与数据稠密区域紧密相连的数据对象划分为一个类，因此与其他对象分离的数据就会作为异常值。也可以使用 K 均值算法实现异常值的检测。首先通过把数据划分为 k 组，划分方式是选择距离各自簇中心最近的点为一组；然后计算每个对象和对应的簇中心的距离（或者相似度），并挑出拥有最大的距离的点作为异常值。

首先对于 x2 变量，即客户的年龄，我们可以定量分析，发现有以下值：

```
unique(traindata$x2)
```

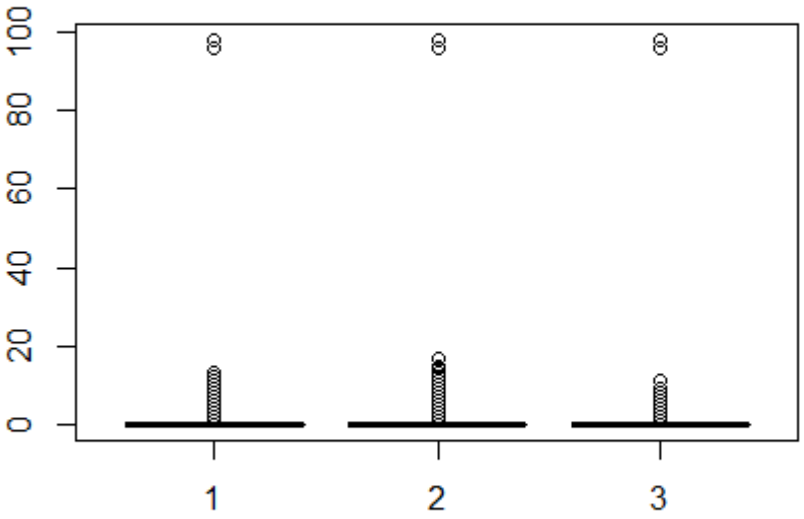
```
##  [1]  45  40  38  30  49  74  57  39  27  51  46  76  64  78  53  43  25
## [18]  32  58  50  69  24  28  62  42  75  26  52  41  81  31  68  70  73
## [35]  29  55  35  72  60  67  36  56  37  66  83  34  44  48  61  80  47
## [52]  59  77  63  54  33  79  65  86  92  23  87  71  22  90  97  84  82
## [69]  91  89  85  88  21  93  96  99  94  95 101  98 103 102 107 105   0
## [86] 109
```

可以看到年龄中存在 0 值，显然是异常值，予以剔除。

```
traindata<-traindata[-which(traindata$x2==0),]
```

而对于 x3,x7,x9 三个变量，由下面的箱线图可以看出，均存在异常值，且由 unique 函数可以得知均存在 96、98 两个异常值，因此予以剔除。同时会发现剔除其中一个变量的 96、98 值，其他变量的 96、98 两个值也会相应被剔除

```
##  [1]  2  0  1  3  4  5  7 10  6 98 12  8  9 96 13 11
##  [1]  0  1  3  2  5  4 98 10  9  6  7  8 15 96 11 13 14 17 12
##  [1]  0  1  2  5  3 98  4  6  7  8 96 11  9
```



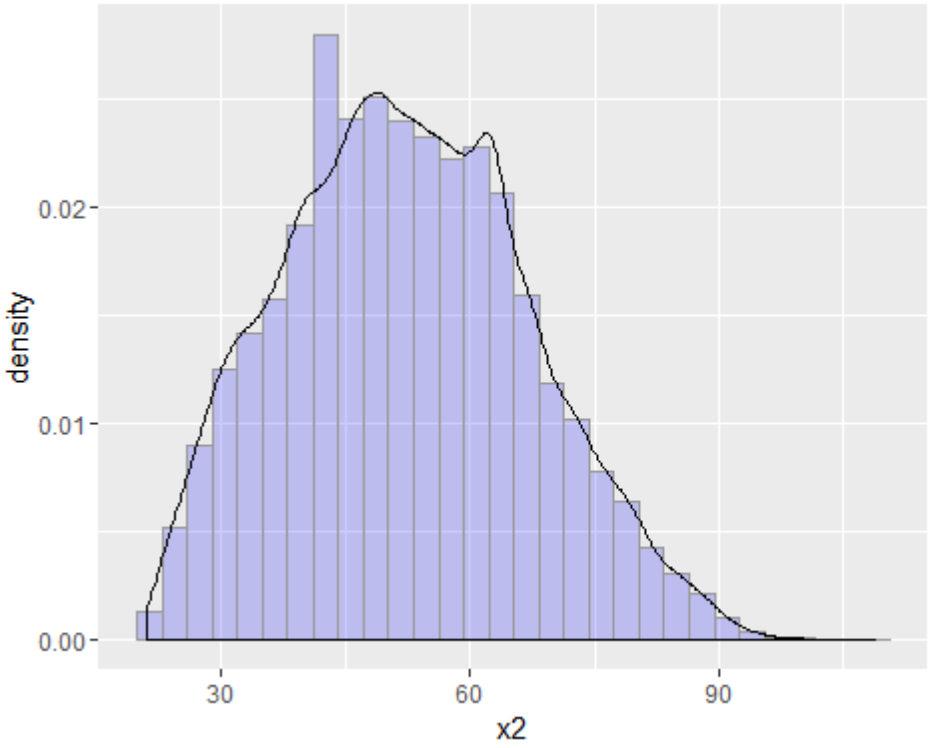
其它变量占不作处理。

三、变量分析

1、单变量分析

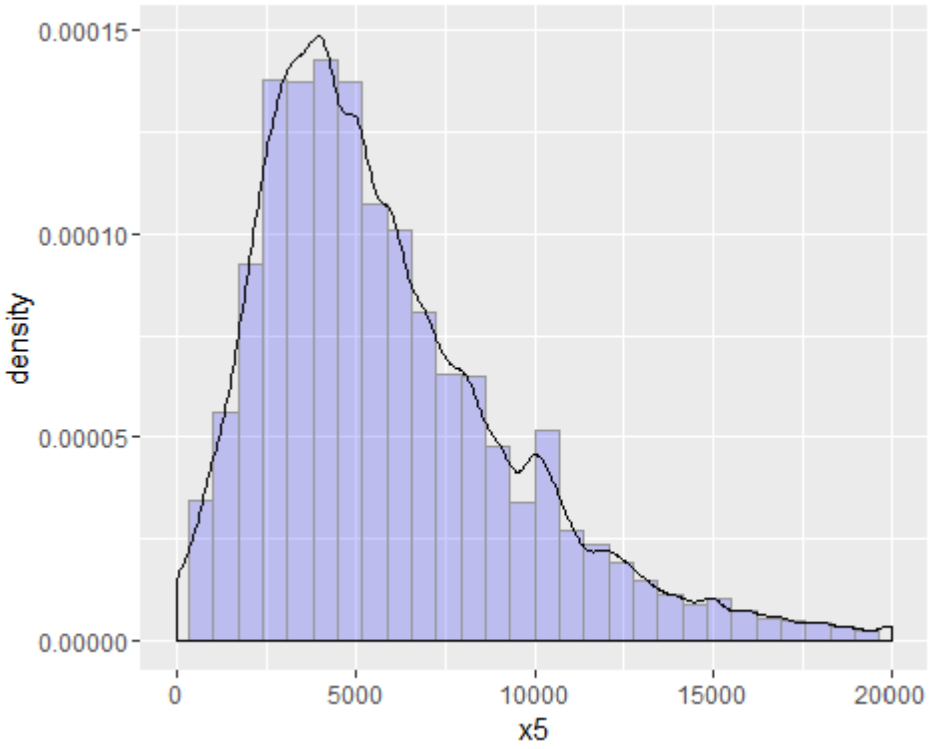
我们可以简单地看下部分变量的分布，比如对于 age 变量，如下图：

```
ggplot(traindata, aes(x = x2, y = ..density..)) + geom_histogram(fill = "blue", colour = "grey60", size = 0.2, alpha = 0.2) + geom_density()
```



可以看到年龄变量大致呈正态分布，符合统计分析的假设。再比如月收入变量，也可以做图观察观察，如下：

```
ggplot(traindata, aes(x = x5, y = ..density..)) + geom_histogram(fill = "blue", colour = "grey60", size = 0.2, alpha = 0.2) + geom_density() + xlim(1, 20000)
```

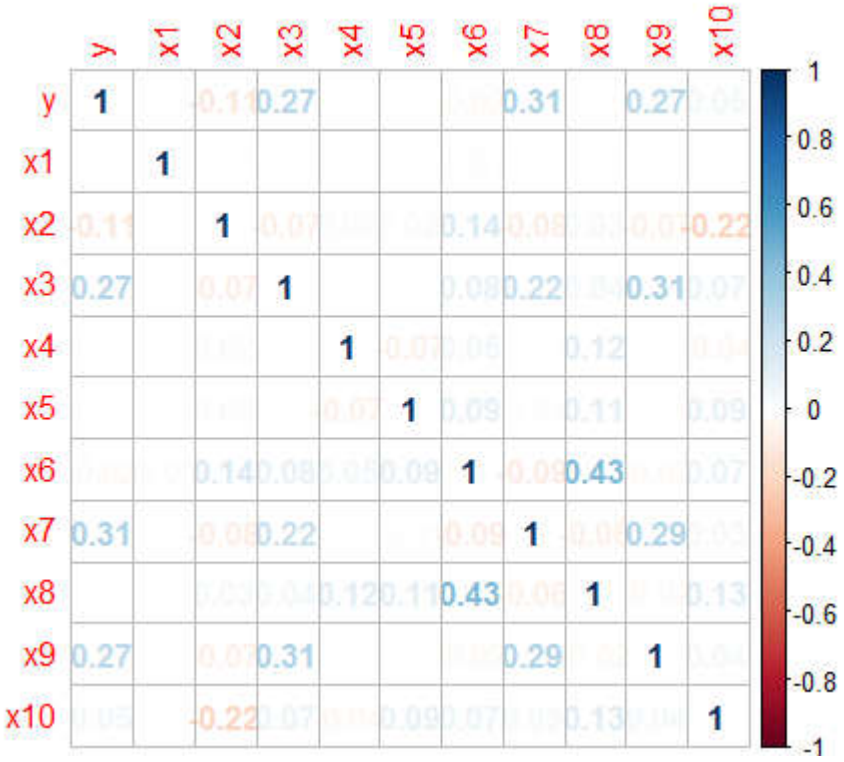


月收入也大致呈正态分布，符合统计分析的需要。

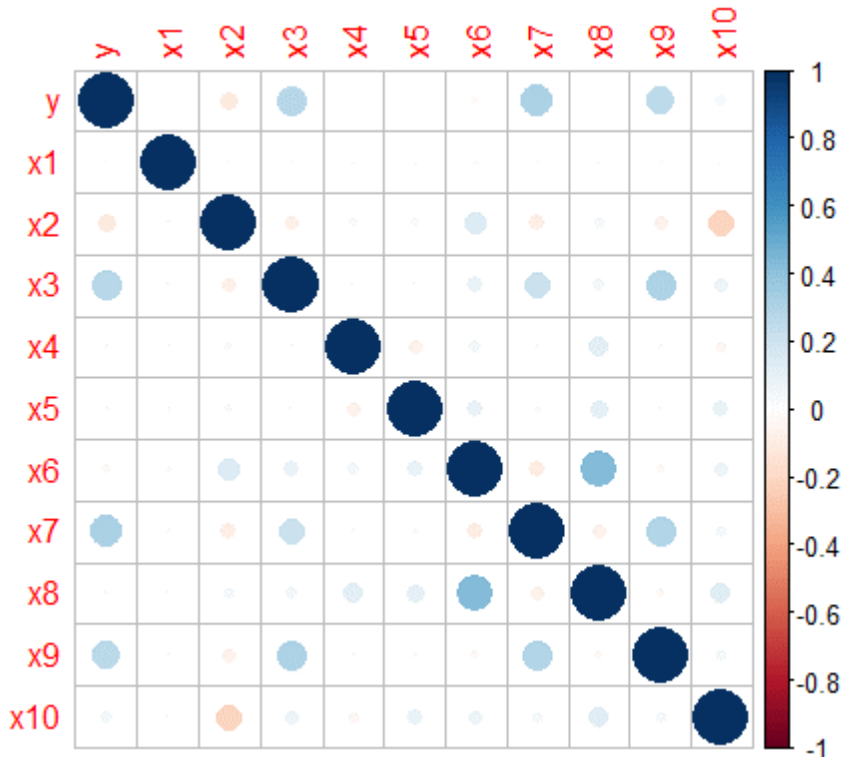
2、变量之间的相关性

建模之前首先得检验变量之间的相关性，如果变量之间相关性显著，会影响模型的预测效果。下面通过 corrplot 函数，画出各变量之间，包括响应变量与自变量的相关性。

```
cor1<-cor(traindata[,1:11])
corrplot(cor1)
```




```
corrplot(cor1,method = "number")
```



由上图可以看出，各变量之间的相关性是非常小的。其实 Logistic 回归同样需要检验多重共线性问题，不过此处由于各变量之间的相关性较小，可以初步判断不存在多重共线性问题，当然我们在建模后还可以用 VIF（方差膨胀因子）来检验多重共线性问题。如果存在多重共线性，即有可能存在两个变量高度相关，需要降维或剔除处理。

四、切分数据集

```
table(traindata$y)
##
##      0      1
## 139851   9879
```

由上表看出，对于响应变量 SeriousDlqin2yrs，存在明显的类失衡问题，SeriousDlqin2yrs 等于 1 的观测为 9879，仅为所有观测值的 6.6%。因此我们需要对非平衡数据进行处理，在这里可以采用 SMOTE 算法，用 R 对稀有事件进行超级采样。

我们利用 caret 包中的 createDataPartition（数据分割功能）函数将数据随机分成相同的两份。

```
set.seed(1234)
splitIndex<-createDataPartition(traindata$y,time=1,p=0.5,list=FALSE)
train<-traindata[splitIndex,]
```

```
test<-traindata[-splitIndex,]
```

对于分割后的训练集和测试集均有 74865 个数据，分类结果的平衡性如下：

```
prop.table(table(train$y))
##
##         0         1
## 0.93314633 0.06685367
prop.table(table(test$y))
##
##         0         1
## 0.93489615 0.06510385
```

两者的分类结果是平衡的，仍然有 6.6%左右的代表，我们仍然处于良好的水平。因此可以采用这份切割的数据进行建模及预测。

五、Logistic 回归

Logistic 回归在信用评分卡开发中起到核心作用。由于其特点，以及对自变量进行了证据权重转换（WOE），Logistic 回归的结果可以直接转换为一个汇总表，即所谓的标准评分卡格式。

1、基本公式

Logistic 回归模型其本身是一个非线性回归模型，经过 logit 转换（连接函数）将相应变量 Y 和线性自变量相联系，可以得到一个线性的形式，使用线性回归模型对参数进行估计，所以说 logistic 回归模型是一个广义线性回归模型。

下面简单地介绍下 Logistic 回归模型。考虑具有 n 个独立变量的向量 $x=(x_1, x_2, \dots, x_n)$，设条件概率 P(y=1|x)=p 为根据观测量相对于某事件 x 发生的概率。那么 Logistic 回归模型可以表示为：

$$P(y=1|x)=\pi(x)=\frac{1}{1+e^{-g(x)}}$$

这里$f(x)=\frac{1}{1+e^{-x}}$称为 Logistic 函数。其中$g(x)=w_0+w_1x_1+\cdots+w_nx_n$，那么在 x 条件下 y 不发生的概率为：

$$P(y=0|x)=1-P(y=1|x)=1-\frac{1}{1+e^{-g(x)}}=\frac{1}{1+e^{g(x)}}$$

所以这个比值称为事件的发生比（the odds of experiencing an event），简记为 odds。对 odds 取对数得到:

$$\ln\left(\frac{P}{1-P}\right)=g(x)=w_0+w_1x_1+\cdots+w_nx_n$$

2、建立模型

首先利用 glm 函数对所有变量进行 Logistic 回归建模，模型如下

```
fit<-glm(y~.,train,family = "binomial")
summary(fit)
##
## Call:
## glm(formula = y ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.6144  -0.3399  -0.2772  -0.2240   3.6997
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.812e+00  6.411e-02 -28.268  < 2e-16 ***
## x1          -1.846e-05  8.972e-05  -0.206 0.836948
```

```
## x2          -2.861e-02  1.276e-03 -22.428  < 2e-16 ***
## x3           5.767e-01  1.564e-02  36.867  < 2e-16 ***
## x4          -2.321e-05  1.538e-05  -1.509 0.131224
## x5          -1.355e-05  3.845e-06  -3.524 0.000425 ***
## x6          -2.769e-03  3.798e-03  -0.729 0.466051
## x7           8.468e-01  2.429e-02  34.855  < 2e-16 ***
## x8           8.620e-02  1.599e-02   5.393 6.94e-08 ***
## x9           8.294e-01  3.338e-02  24.848  < 2e-16 ***
## x10          5.126e-02  1.388e-02   3.694 0.000221 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 36747  on 74864  degrees of freedom
## Residual deviance: 29793  on 74854  degrees of freedom
## AIC: 29815
##
## Number of Fisher Scoring iterations: 6
```

可以看出，利用全变量进行回归，模型拟合效果并不是很好，其中 x1,x4,x6 三个变量的 p 值未能通过检验，在此直接剔除这三个变量，利用剩余的变量对 y 进行回归。

```
fit2<-glm(y~x2+x3+x5+x7+x8+x9+x10,train,family = "binomial")
summary(fit2)
##
## Call:
## glm(formula = y ~ x2 + x3 + x5 + x7 + x8 + x9 + x10, family = "binomial",
##     data = train)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -4.6223  -0.3402  -0.2777  -0.2239   3.5868
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.825e+00  6.320e-02 -28.873  < 2e-16 ***
## x2          -2.894e-02  1.252e-03 -23.120  < 2e-16 ***
## x3           5.742e-01  1.544e-02  37.187  < 2e-16 ***
## x5          -1.185e-05  3.513e-06  -3.373 0.000744 ***
## x7           8.500e-01  2.401e-02  35.397  < 2e-16 ***
## x8           7.494e-02  1.420e-02   5.276 1.32e-07 ***
## x9           8.306e-01  3.338e-02  24.883  < 2e-16 ***
## x10          5.169e-02  1.386e-02   3.730 0.000192 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 36747  on 74864  degrees of freedom
## Residual deviance: 29797  on 74857  degrees of freedom
## AIC: 29813
##
## Number of Fisher Scoring iterations: 6
```

第二个回归模型所有变量都通过了检验，甚至 AIC 值（赤池信息准则）更小，所有模型的拟合效果更好些。

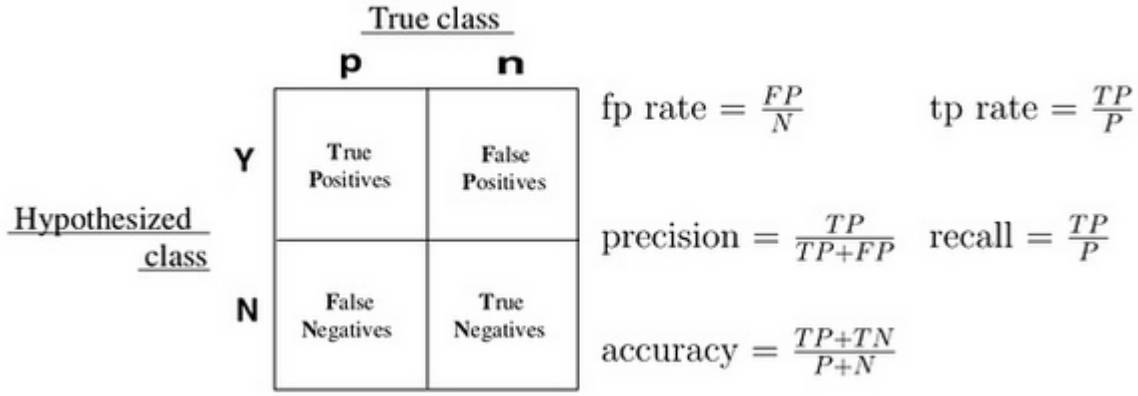
3、模型评估

通常一个二值分类器可以通过 ROC（Receiver Operating Characteristic）曲线和 AUC 值来评价优劣。

很多二元分类器会产生一个概率预测值，而非仅仅是 0-1 预测值。我们可以使用某个临界点（例如 0.5），以划分哪些预测为 1，哪些预测为 0。得到二元预测值后，可以构建一个混淆矩阵来评价二元分类器的预测效果。所有的训练数据都会落入这个矩阵中，而对角线上的数字代表了预测正确的数目，即 true positive + true nagetive。同时可以相应算出 TPR（真正率或称为灵敏度）和 TNR（真负率或称为特异度）。我们主观上希望这两个指标越大越好，但可惜二者是一个此消彼涨的关系。除了分类器的训练参数，临界点的选择，也会大大的影响 TPR 和 TNR。有时可以根据具体问题和需要，来选择具体的临界点。

如果我们选择一系列的临界点，就会得到一系列的 TPR 和 TNR，将这些值对应的点连接起来，就构成了 ROC 曲线。ROC 曲线可以帮助我们清楚的了解到这个分类器的性能表现，还能方便比较不同分类器的性能。在绘制 ROC 曲线的时候，习惯上是使用 1-TNR 作为横坐标即 FPR（false positive rate），TPR 作为纵坐标。这是就形成了 ROC 曲线。

而 AUC（Area Under Curve）被定义为 ROC 曲线下的面积，显然这个面积的数值不会大于 1。又由于 ROC 曲线一般都处于 y=x 这条直线的上方，所以 AUC 的取值范围在 0.5 和 1 之间。使用 AUC 值作为评价标准是因为很多时候 ROC 曲线并不能清晰的说明哪个分类器的效果更好，而作为一个数值，对应 AUC 更大的分类器效果更好。



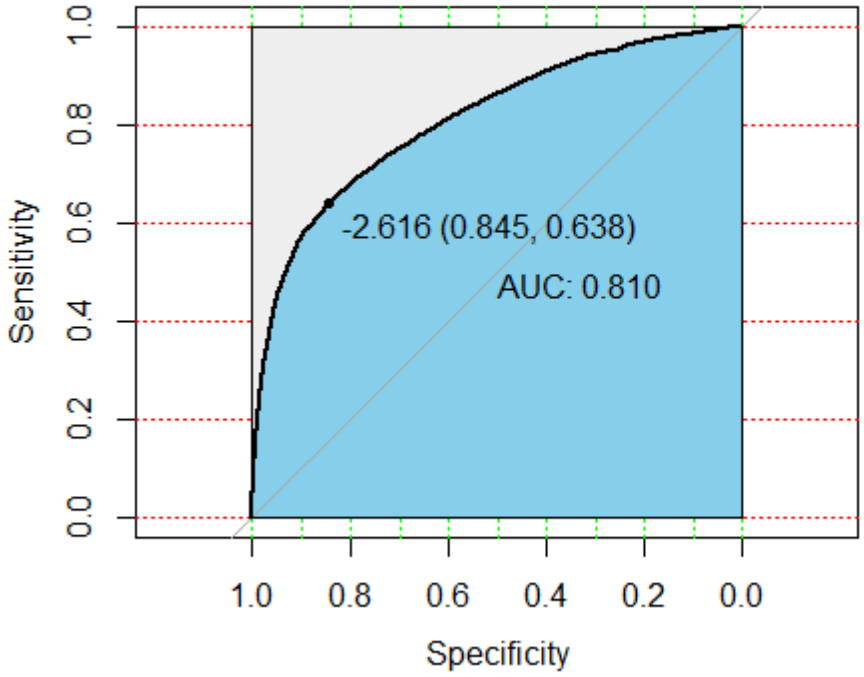
下面首先利用模型对 test 数据进行预测，生成概率预测值

```
pre <- predict(fit2,test)
```

在 R 中，可以利用 pROC 包，它能方便比较两个分类器，还能自动标注出最优的临界点，图看起来也比较漂亮。在下图中最优点 FPR=1-TNR=0.845，TPR=0.638，AUC 值为 0.8102，说明该模型的预测效果还是不错的，正确较高。

```
modelroc <- roc(test$y,pre)
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="skyblue", print.thres=TRUE)
```




```
##
## Call:
## roc.default(response = test$y, predictor = pre)
##
## Data: pre in 69991 controls (test$y 0) < 4874 cases (test$y 1).
## Area under the curve: 0.8102
```

六、WOE 转换

证据权重（Weight of Evidence,WOE）转换可以将 Logistic 回归模型转变为标准评分卡格式。引入 WOE 转换的目的并不是为了提高模型质量，只是一些变量不应该被纳入模型，这或者是因为它们不能增加模型值，或者是因为与其模型相关系数有关的误差较大，其实建立标准信用评分卡也可以不采用 WOE 转换。这种情况下，Logistic 回归模型需要处理更大数量的自变量。尽管这样会增加建模程序的复杂性，但最终得到的评分卡都是一样的。

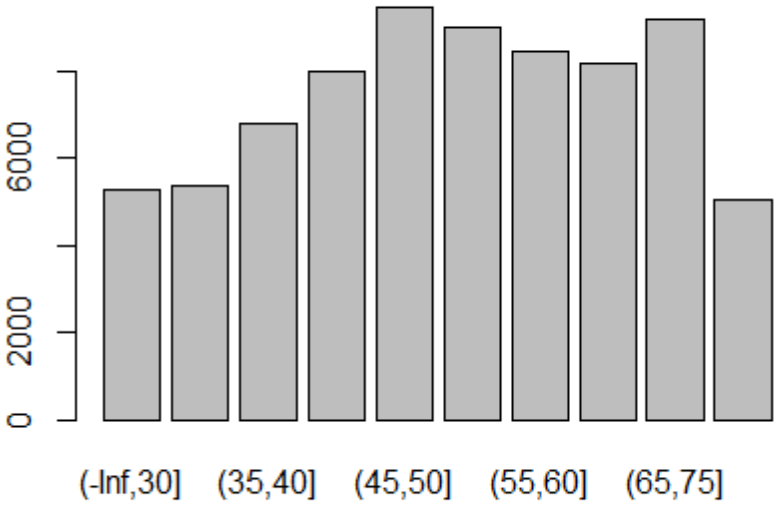
用 WOE(x)替换变量 x。WOE()=ln[(违约/总违约)/(正常/总正常)]。

通过上述的 Logistic 回归，剔除 x1,x4,x6 三个变量，对剩下的变量进行 WOE 转换。

1、进行分箱

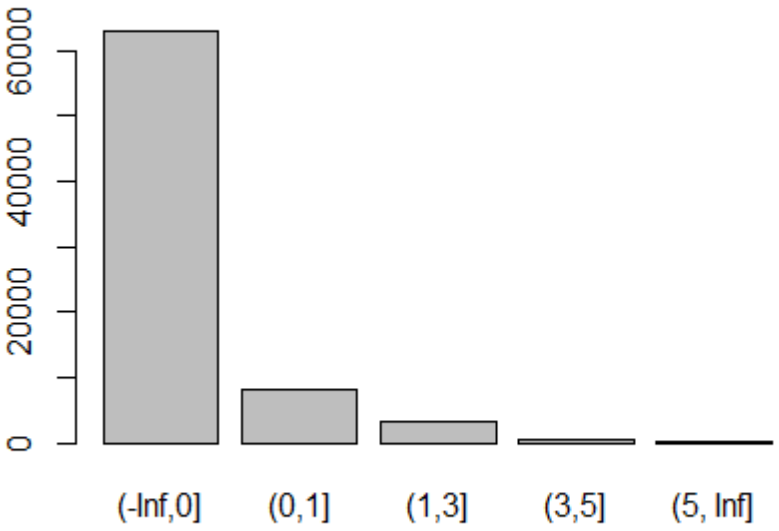
age 变量(x2)：

```
cutx2= c(-Inf,30,35,40,45,50,55,60,65,75,Inf)
plot(cut(train$x2,cutx2))
```



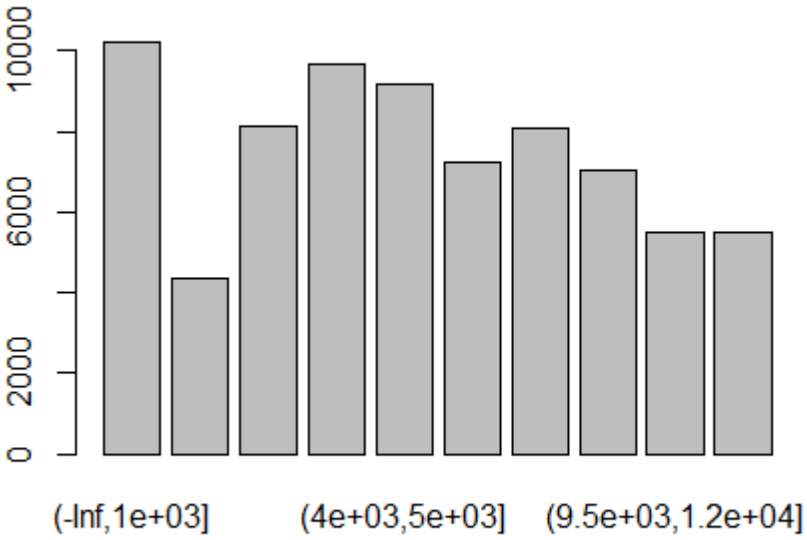
NumberOfTime30-59DaysPastDueNotWorse 变量(x3)：

```
cutx3 = c(-Inf,0,1,3,5,Inf)
plot(cut(train$x3,cutx3))
```



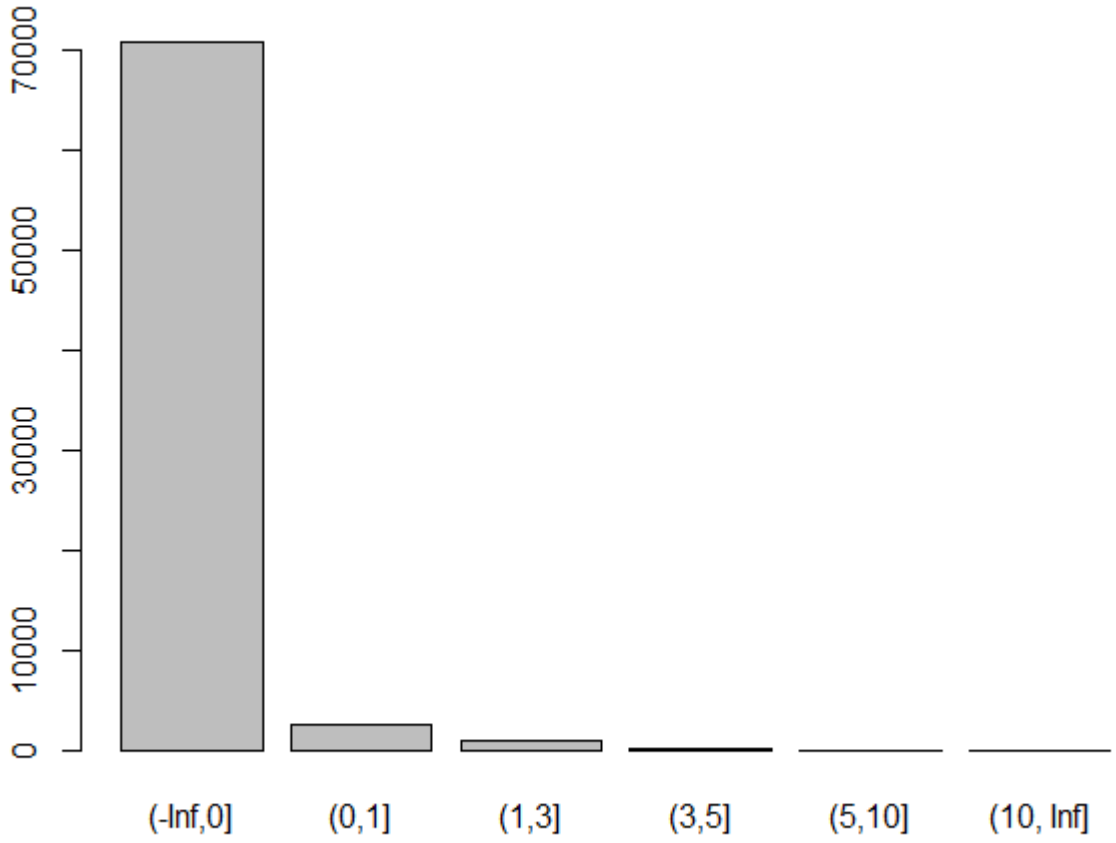
MonthlyIncome 变量(x5)：

```
cutx5 = c(-Inf,1000,2000,3000,4000,5000,6000,7500,9500,12000,Inf)
plot(cut(train$x5,cutx5))
```



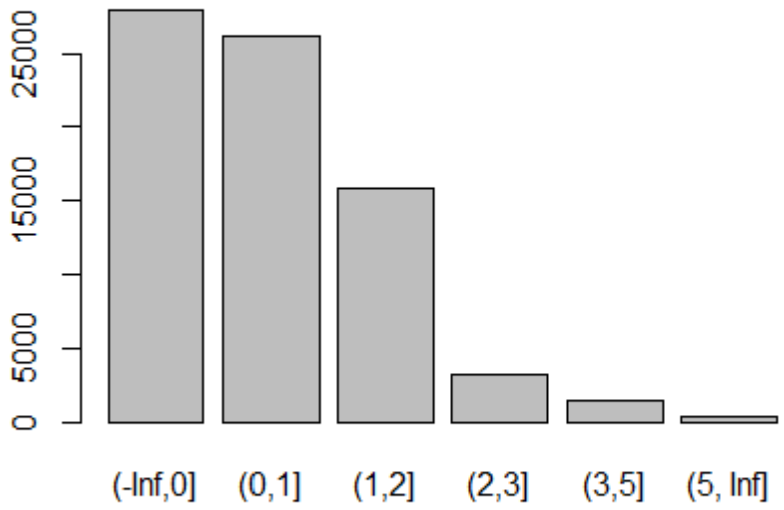
NumberOfTimes90DaysLate 变量(x7)：

```
cutx7 = c(-Inf,0,1,3,5,10,Inf)
plot(cut(train$x7,cutx7))
```



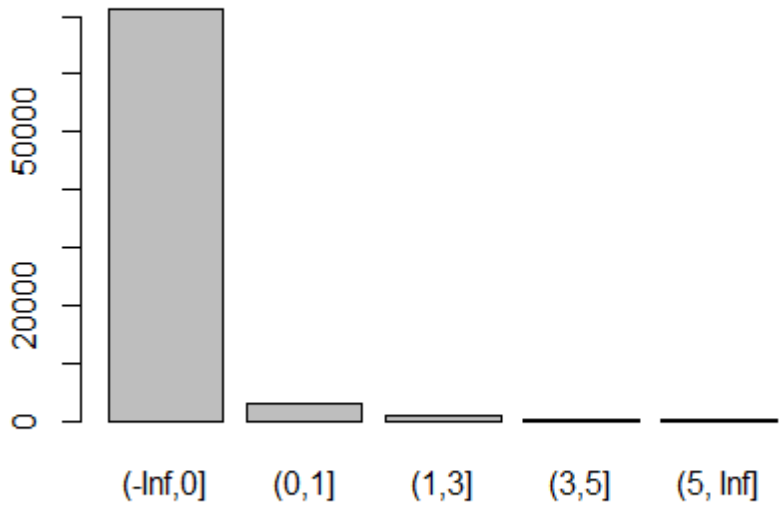
NumberRealEstateLoansOrLines 变量(x8)：

```
cutx8= c(-Inf,0,1,2,3,5,Inf)
plot(cut(train$x8,cutx8))
```



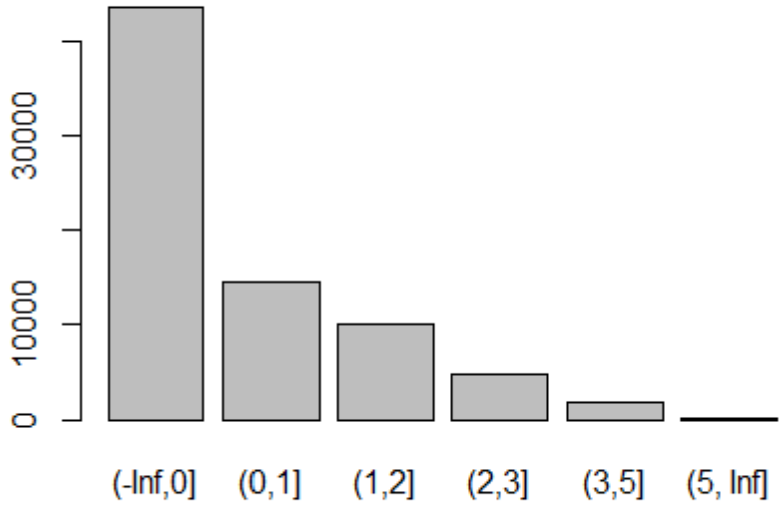
NumberOfTime60-89DaysPastDueNotWorse 变量(x9)：

```
cutx9 = c(-Inf,0,1,3,5,Inf)
plot(cut(train$x9,cutx9))
```



NumberOfDependents 变量(x10)：

```
cutx10 = c(-Inf,0,1,2,3,5,Inf)
plot(cut(train$x10,cutx10))
```



2、计算 WOE 值

计算 WOE 的函数

```
totalgood = as.numeric(table(train$y))[1]
totalbad = as.numeric(table(train$y))[2]
getWOE <- function(a,p,q)
{
  Good <- as.numeric(table(train$y[a > p & a <= q]))[1]
  Bad <- as.numeric(table(train$y[a > p & a <= q]))[2]
  WOE <- log((Bad/totalbad)/(Good/totalgood),base = exp(1))
  return(WOE)
}
```

比如 age 变量(x2)

```
Agelessthan30.WOE=getWOE(train$x2,-Inf,30)
```

```
  Age30to35.WOE=getWOE(train$x2,30,35)
  Age35to40.WOE=getWOE(train$x2,35,40)
  Age40to45.WOE=getWOE(train$x2,40,45)
  Age45to50.WOE=getWOE(train$x2,45,50)
  Age50to55.WOE=getWOE(train$x2,50,55)
  Age55to60.WOE=getWOE(train$x2,55,60)
  Age60to65.WOE=getWOE(train$x2,60,65)
  Age65to75.WOE=getWOE(train$x2,65,75)
  Agemorethan.WOE=getWOE(train$x2,75,Inf)

age.WOE=c(Agelessthan30.WOE,Age30to35.WOE,Age35to40.WOE,Age40to45.WOE,
Age45to50.WOE,

Age50to55.WOE,Age55to60.WOE,Age60to65.WOE,Age65to75.WOE,Agemorethan.W
OE)
  age.WOE
## [1] 0.57432879 0.52063157 0.34283924 0.24251193 0.22039521
## [6] 0.07194294 -0.25643603 -0.55868003 -0.94144504 -1.28914527
```

NumberOfTime30-59DaysPastDueNotWorse 变量(x3)

```
## [1] -0.5324915 0.9106018 1.7645290 2.4432903 2.5682332
```

MonthlyIncome 变量(x5)

```
## [1] -1.128862326 0.448960482 0.312423080 0.350846777 0.247782295
## [6] 0.114417168 -0.001808106 -0.237224039 -0.389158800 -0.462438653
```

NumberOfTimes90DaysLate 变量(x7)

```
## [1] -0.3694044 1.9400973 2.7294448 3.3090003 3.3852925 2.3483738
```

NumberRealEstateLoansOrLines 变量(x8)

```
## [1] 0.21490691 -0.24386987 -0.15568385 0.02906876 0.41685234
1.12192809
```

NumberOfTime60-89DaysPastDueNotWorse 变量(x9)

```
## [1] -0.2784605 1.8329078 2.7775343 3.5805174 3.4469860
```

NumberOfDependents 变量(x10)

[1] -0.15525081 0.08669961 0.19618098 0.33162486 0.40469824
0.76425365

3、对变量进行 WOE 变换

如 age 变量(x2)

如 age 变量(x2)

```
tmp.age <- 0
for(i in 1:nrow(train)) {
  if(train$x2[i] <= 30)
    tmp.age[i] <- Agelessthan30.WOE
  else if(train$x2[i] <= 35)
    tmp.age[i] <- Age30to35.WOE
  else if(train$x2[i] <= 40)
    tmp.age[i] <- Age35to40.WOE
  else if(train$x2[i] <= 45)
    tmp.age[i] <- Age40to45.WOE
  else if(train$x2[i] <= 50)
    tmp.age[i] <- Age45to50.WOE
  else if(train$x2[i] <= 55)
    tmp.age[i] <- Age50to55.WOE
  else if(train$x2[i] <= 60)
    tmp.age[i] <- Age55to60.WOE
  else if(train$x2[i] <= 65)
    tmp.age[i] <- Age60to65.WOE
  else if(train$x2[i] <= 75)
    tmp.age[i] <- Age65to75.WOE
  else
    tmp.age[i] <- Agemorethan.WOE
}

table(tmp.age)
```

```
## tmp.age
##   -1.2891452711972 -0.941445039519045 -0.558680027962495
##               5063               9196               8180
## -0.256436029353835 0.0719429392949312  0.220395209955515
##               8472               9009               9465
##  0.242511934081286  0.342839240194068   0.52063156705216
##               8008               6784               5390
##  0.574328792863984
##               5298
   tmp.age[1:10]
##  [1] 0.34283924 0.57432879 0.34283924 0.57432879 0.07194294 0.22039521
##  [7] 0.07194294 0.24251193 0.34283924 0.52063157
   train$x2[1:10]
##  [1] 38 30 39 30 51 46 53 43 39 32
```

NumberOfTime30-59DaysPastDueNotWorse 变量(x3)

```
## tmp.NumberOfTime30.59DaysPastDueNotWorse
## -0.53249146131578 0.910601840444591  1.76452904024992
2.44329031065646
##             62948              8077              3160               562
##  2.56823323027274
##               118
##  [1]  0.9106018 -0.5324915 -0.5324915 -0.5324915 -0.5324915 -0.5324915
##  [7] -0.5324915 -0.5324915 -0.5324915 -0.5324915
##  [1] 1 0 0 0 0 0 0 0 0 0
```

MonthIncome 变量(x5)

```
## tmp.MonthlyIncome
##    -1.12886232582259    -0.462438653207328    -0.389158799506996
##                10201                 5490                 5486
##   -0.237224038650003 -0.00180810632297072     0.114417167554772
##                 7048                 8076                 7249
```

```
##    0.247782294610166    0.312423079500641    0.350846777249291
##                 9147                 8118                 9680
##    0.448960482499888
##                 4370
##  [1]  0.350846777  0.350846777  0.350846777  0.312423080 -0.001808106
##  [6] -0.462438653 -0.237224039  0.350846777  0.312423080 -0.237224039
##  [1]  3042  3300  3500  2500  6501 12454  8800  3280  2500  7916
```

NumberOfTime90DaysPastDueNotWorse 变量(x7)

```
## tmp.NumberOfTimes90DaysLate
## -0.369404425455224    1.94009728631401    2.34837375415972
##              70793               2669                  7
##   2.72944477623793    3.30900029985393    3.38529247382249
##               1093                222                 81
##  [1]  1.9400973 -0.3694044 -0.3694044 -0.3694044 -0.3694044 -0.3694044
##  [7] -0.3694044 -0.3694044 -0.3694044 -0.3694044
##  [1] 1 0 0 0 0 0 0 0 0 0
```

NumberRealEstateLoansOrLines 变量(x8)

```
## tmp.NumberRealEstateLoansOrLines
## -0.243869874062293 -0.155683851792327 0.0290687559545721
##              26150              15890               3130
##  0.214906905417014    1.12192809398173
##              27901               1794
##  [1]  0.2149069  0.2149069  0.2149069  0.2149069 -0.1556839 -0.1556839
##  [7]  0.2149069 -0.2438699  0.2149069  0.2149069
##  [1] 0 0 0 0 2 2 0 1 0 0
```

NumberOfTime60.89DaysPastDueNotWorse 变量(x9)

```
## tmp.NumberOfTime60.89DaysPastDueNotWorse
## -0.278460464730538    1.83290775083723    2.77753428092856
##              71150               2919                708
```

```
##   3.44698604282783   3.58051743545235
##                13                75
##  [1] -0.2784605 -0.2784605 -0.2784605 -0.2784605 -0.2784605 -0.2784605
##  [7] -0.2784605 -0.2784605 -0.2784605 -0.2784605
##  [1] 0 0 0 0 0 0 0 0 0 0
```

NumberOfDependents 变量(x10)

```
## tmp.NumberOfDependents
## -0.155250809857344 0.0866996065110081  0.196180980387687
##              43498             14544             10102
##  0.331624863227172  0.404698242905824   0.76425364970991
##               4771              1815               135
##  [1] -0.1552508 -0.1552508 -0.1552508 -0.1552508  0.1961810  0.1961810
##  [7] -0.1552508  0.1961810 -0.1552508 -0.1552508
##  [1] 0 0 0 0 2 2 0 2 0 0
```

4、WOE DataFrame 构建：

```
trainWOE
=cbind.data.frame(tmp.age,tmp.NumberOfTime30.59DaysPastDueNotWorse,tmp.MonthlyIncome,tmp.NumberOfTime60.89DaysPastDueNotWorse,tmp.NumberOfTimes90DaysLate,tmp.NumberRealEstateLoansOrLines,tmp.NumberOfDependents)
```

七、评分卡的创建和实施

标准评分卡采用的格式是评分卡中的每一个变量都遵循一系列 IF-THEN 法则，变量的值决定了该变量所分配的分值，总分就是各变量分值的和。

评分卡设定的分值刻度可以通过将分值表示为违约和正常概率比对数(log(odds))的线性表达式来定义：

$$\text{Score} = \text{A} - \text{Blog}(\text{Odds}) \qquad Odds = \frac{p}{1-p}$$

在前面的流程中我们得到了每个变量经 WOE 变换后各类别的 WOE 值和各变量的 logistic 回归模型的系数（$\beta_0, \beta_1 \ldots$），那么每条记录的违约和正常概率比的对数(log(odds))就可以得到。

$$Score = A - B(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p)$$

$$Score = A - B\begin{Bmatrix} \beta_0 \\ +(\beta_1 w_{11})\delta_{11} + (\beta_1 w_{12})\delta_{12} + \cdots \\ \cdots \\ +(\beta_p w_{p1})\delta_{p1} + (\beta_p w_{p2})\delta_{p2} + \cdots \end{Bmatrix}$$

$$\begin{aligned} Score = {} & (A - B\beta_0) \\ & - (B\beta_1 w_{11})\delta_{11} - (B\beta_1 w_{12})\delta_{12} - \cdots \\ & - \cdots \\ & - (B\beta_p w_{p1})\delta_{p1} - (B\beta_p w_{p2})\delta_{p2} - \cdots \end{aligned}$$

知道线性表达式的两个参数 A，B 后就可以求每条记录（申请人）的分值。为了求得 A，B，需要设定两个假设（分数的给定，很主观）。

以上就是推断，实际代码中，习惯用了 q、p 来代表 A、B.

通俗来说就是，评分需要自己预设一个阀值，比如：

这个人预测出来"不发生违约"的几率为 0.8，设定这个人为 500 分；

另一个人预测出来"不发生违约"的几率为 0.9，设定这个人为 600 分。

阀值的设定需根据行业经验不断跟踪调整，下面的分数设定仅代表个人经验。

下面开始设立评分，假设按好坏比 15 为 600 分，每高 20 分好坏比翻一倍算出 P,Q。
如果后期结果不明显，可以高 30-50 分好坏比才翻一倍。

Score = q - p * log(odds)

即有方程：

620 = q - p * log(15)

600 = q - p * log(15/2)

逻辑回归建模：

#因为数据中"1"代表的是违约，直接建模预测，求的是"发生违约的概率"，log(odds)即为"坏好比"。为了符合常规理解，分数越高，信用越好，所有就调换"0"和"1"，使建模预测结果为"不发生违约的概率"，最后 log(odds)即表示为"好坏比"。

```
trainWOE$y = 1-train$y

glm.fit = glm(y~.,data = trainWOE,family = binomial(link = logit))

summary(glm.fit)

coe = (glm.fit$coefficients)

p <- 20/log(2)

q <- 600-20*log(15)/log(2)

Score=q + p*{as.numeric(coe[1])+as.numeric(coe[2])*tmp.age
+as.numeric(coe[3])*tmp.NumberOfTime30.59DaysPastDueNotWorse+p*as.numeric
(coe[4])*tmp.MonthlyIncome+p*as.numeric(coe[5])*tmp.NumberOfTime60.89DaysP
astDueNotWorse+p*as.numeric(coe[6])*tmp.NumberOfTimes90DaysLate+p*as.num
eric(coe[7])*tmp.NumberRealEstateLoansOrLines+p*as.numeric(coe[8])*tmp.Numbe
rOfDependents
```

个人总评分=基础分+各部分得分

基础分为:

```
  base <- q + p*as.numeric(coe[1])

  base
```

```
## [1] 446.2841
```

1、对各变量进行打分

比如 age 变量(x2)

```
   Agelessthan30.SCORE = p*as.numeric(coe[2])*Agelessthan30.WOE

   Age30to35.SCORE = p*as.numeric(coe[2])*Age30to35.WOE

   Age35to40.SCORE = p*as.numeric(coe[2])*Age35to40.WOE

   Age40to45.SCORE = p*as.numeric(coe[2])*Age40to45.WOE

   Age45to50.SCORE = p*as.numeric(coe[2])*Age45to50.WOE

   Age50to55.SCORE = p*as.numeric(coe[2])*Age50to55.WOE
```

```
    Age55to60.SCORE = p*as.numeric(coe[2])*Age55to60.WOE

    Age60to65.SCORE = p*as.numeric(coe[2])*Age60to65.WOE

    Age65to75.SCORE = p*as.numeric(coe[2])*Age65to75.WOE

    Agemorethan.SCORE=p*as.numeric(coe[2])*Agemorethan.WOE

    Age.SCORE
=c(Age30to35.SCORE,Age35to40.SCORE,Age40to45.SCORE,Age45to50.SCORE,Age5
0to55.SCORE,Age55to60.SCORE,Age60to65.SCORE,Age65to75.SCORE,Agemorethan.
SCORE)

    Age.SCORE

## [1]  10.498828   6.913546   4.890389   4.444393   1.450770  -5.171176

## [7] -11.266096 -18.984767 -25.996338
```

NumberOfTime30-59DaysPastDueNotWorse 变量(x3)

```
## [1] -10.29843  17.61112  34.12614  47.25344  49.66985
```

MonthlyIncome 变量(x5)

```
##  [1] -24.92797904   9.91412083   6.89904854   7.74753565   5.47162546

##  [6]   2.52660461  -0.03992731  -5.23847393  -8.59355669 -10.21175106
```

NumberOfTimes90DaysLate 变量(x7)

```
## [1] -5.19482 27.28299 38.38333 46.53344 47.60632 33.02445
```

NumberRealEstateLoansOrLine 变量(x8)

```
## [1]  4.022310 -4.564396 -2.913860  0.544066  7.802025 20.998590
```

NumberOfTime60-89DaysPastDueNotWorse 变量(x9)

```
## [1] -4.820833 31.732126 48.085927 61.987533 59.675778
```

NumberOfDependents 变量(x10)

```
## [1] -1.5734012  0.8786638  1.9882112  3.3608775  4.1014453  7.7453871
```

构造计算分值函数：

```
  getscore<-function(i,x){

  score = round(p*as.numeric(coe[i])*x,0)

  return(score)

}
```

2、计算各变量分箱得分：

age 变量(x2)

```
    Agelessthan30.SCORE = getscore(2,Agelessthan30.WOE)

    Age30to35.SCORE = getscore(2,Age30to35.WOE)

    Age35to40.SCORE = getscore(2,Age35to40.WOE)

    Age40to45.SCORE = getscore(2,Age40to45.WOE)

    Age45to50.SCORE = getscore(2,Age45to50.WOE)

    Age50to55.SCORE = getscore(2,Age50to55.WOE)

    Age55to60.SCORE = getscore(2,Age55to60.WOE)

    Age60to65.SCORE = getscore(2,Age60to65.WOE)

    Age65to75.SCORE = getscore(2,Age65to75.WOE)

    Agemorethan.SCORE = getscore(2,Agemorethan.WOE)

    Age.SCORE =
c(Agelessthan30.SCORE,Age30to35.SCORE,Age35to40.SCORE,Age40to45.SCORE,Age
45to50.SCORE,Age50to55.SCORE,Age55to60.SCORE,Age60to65.SCORE,Age65to75.S
CORE,Agemorethan.SCORE)

    Age.SCORE

##  [1]  12  10   7   5   4   1  -5 -11 -19 -26
```

NumberOfTime30-59DaysPastDueNotWorse 变量(x3)

```
## [1] -10  18  34  47  50
```

MonthlyIncome 变量(x5)

```
##  [1] -25  10   7   8   5   3   0   0  -9 -10
```

NumberOfTimes90DaysLate 变量(x7)

```
## [1] -5 27 38 47 48 33
```

NumberRealEstateLoansOrLine 变量(x8)

```
## [1]  4 -5 -3  1  8 21
```

NumberOfTime60-89DaysPastDueNotWorse 变量(x9)

```
## [1] -5 32 48 62 60
```

NumberOfDependents 变量(x10)

[1] -2 1 2 3 4 8

3、最终生成的评分卡如下：

age	X2	<=30	(30,35]	(35,40]	(40,45]	(45,50]	(50,55]
	Score	12	10	7	5	4	1
NumberOfTime30-59DaysPastDueNotWorse	X3	<=0	(0,1]	(1,3]	(3,5]	>5	
	Score	-10	18	34	47	50	
MonthlyIncome	X5	<=1000	(1000,2000]	(2000,3000]	(3000,4000]	(4000,5000]	(5000,60
	Score	-25	10	7	8	6	3
NumberOfTimes90DaysLate	X7	<=0	(0,1]	(1,3]	(3,5]	(5,10]	>10
	Score	-5	27	38	47	48	33
NumberRealEstateLoansOrLines	X8	<=0	(0,1]	(1,2]	(2,3]	(3,5]	>5
	Score	4	-5	-3	1	8	21
NumberOfTime60-89DaysPastDueNotWorse	X9	<=0	(0,1]	(1,3]	(3,5]	>5	
	Score	-5	32	48	62	60	
NumberOfDependents	X10	<=0	(0,1]	(1,2]	(2,3]	(3,5]	>5
	Score	-2	1	2	3	4	8

个人评分计算案例：

特征	数据	分数
Age	38	7
NumberOfTime30-59DaysPastDueNotWorse	4	47
MonthlyIncome	1500	10
NumberOfTimes90DaysLate	2	38
NumberRealEstateLoansOrLines	1.5	-3
NumberOfTime60-89DaysPastDueNotWorse	4	62
NumberOfDependents	1.5	2

所以这个人的总评分=基础分（base）+各特征分数

总评分=446.2841+7+47+10+38-3+62+2=609.2841