使用R语言开发评分卡模型

scorecard包简介

谢士晨

2017年11月

主要内容

1. 评分卡是什么？

2. R语言环境配置

3. scorecard包的使用

1. 评分卡是什么？

评分卡相关概念*

- **定义：** 信用评分模型运用数据挖掘技术和统计分析方法，通过对消费者的人口特征、信用历史记录、行为记录、交易记录等大量数据进行系统的分析，挖掘数据中蕴含的行为模式、信用特征，捕捉历史信息和未来信用表现之间的关系，发展出预测性的模型，以一个信用评分来综合评估消费者未来的某种信用表现。

评分卡相关概念*

- **定义：** 信用评分模型运用数据挖掘技术和统计分析方法，通过对消费者的人口特征、信用历史记录、行为记录、交易记录等大量数据进行系统的分析，挖掘数据中蕴含的行为模式、信用特征，捕捉历史信息和未来信用表现之间的关系，发展出预测性的模型，以一个信用评分来综合评估消费者未来的某种信用表现。

- **信用评分模型分类：**

  - 金融机构内部数据评分
    - 申请风险评分
    - 行为风险评分、行为收益评分、行为流失倾向评分
    - 申请欺诈评分、交易欺诈评分
  - 征信局数据评分
    - 征信局风险/破产/收益评分、市场反应评分、转账倾向评分

评分卡相关概念*

- **定义：** 信用评分模型运用数据挖掘技术和统计分析方法，通过对消费者的人口特征、信用历史记录、行为记录、交易记录等大量数据进行系统的分析，挖掘数据中蕴含的行为模式、信用特征，捕捉历史信息和未来信用表现之间的关系，发展出预测性的模型，以一个信用评分来综合评估消费者未来的某种信用表现。
- **信用评分模型分类：**
  - 金融机构内部数据评分
    - 申请风险评分
    - 行为风险评分、行为收益评分、行为流失倾向评分
    - 申请欺诈评分、交易欺诈评分
  - 征信局数据评分
    - 征信局风险/破产/收益评分、市场反应评分、转账倾向评分
- **信用评分模型优点：** 客观性、准确性、效率性

评分卡相关概念*

- **定义：** 信用评分模型运用数据挖掘技术和统计分析方法，通过对消费者的人口特征、信用历史记录、行为记录、交易记录等大量数据进行系统的分析，挖掘数据中蕴含的行为模式、信用特征，捕捉历史信息和未来信用表现之间的关系，发展出预测性的模型，以一个信用评分来综合评估消费者未来的某种信用表现。
- **信用评分模型分类：**
  - 金融机构内部数据评分
    - 申请风险评分
    - 行为风险评分、行为收益评分、行为流失倾向评分
    - 申请欺诈评分、交易欺诈评分
  - 征信局数据评分
    - 征信局风险/破产/收益评分、市场反应评分、转账倾向评分
- **信用评分模型优点：** 客观性、准确性、效率性

[*] 陈建. 信用评分模型技术与应用[M]. 中国财政经济出版社, 2005.

标准评分卡示例

variable	bin	points
basepoints	NA	449
age.in.years	[-Inf,26)	-17
age.in.years	[26,35)	0
age.in.years	[35,40)	20
age.in.years	[40, Inf)	2
housing	own	17
housing	rent	-25
housing	for free	-48
present.employment.since	unemployed%,%... < 1 year	-23
present.employment.since	1 <= ... < 4 years	-4
present.employment.since	4 <= ... < 7 years	24
present.employment.since	... >= 7 years	12

标准评分卡示例（续）

- 如果某进件客户信息如下：

present.employment.since	housing	age.in.years
1 <= ... < 4 years	rent	35

- 则相应信用评分为：

age.in.years	housing	present.employment.since	score
-17	-25	-4	403

2. R语言环境配置

软件安装与配置

- 下载软件: R(CRAN/CRAN清华镜像,国内速度更快), IDE(Rstudio)

软件安装与配置

- 下载软件: R(CRAN/CRAN清华镜像,国内速度更快), IDE(Rstudio)
- 自定义启动环境*:
  - R启动时在用户主目录中寻找`.Rprofile`文件，并自动执行其中的R代码。
  - 用户主目录路径可通过`Sys.getenv("HOME")`获取或者刚打开R时通过`getwd()`获得。
  - 在R中执行`file.edit('~/.Rprofile')`，然后将相关代码(见下页)写入并保存。

软件安装与配置

- **下载软件**: R(CRAN/CRAN清华镜像,国内速度更快), IDE(Rstudio)
- **自定义启动环境**[*]:
  - R启动时在用户主目录中寻找`.Rprofile`文件，并自动执行其中的R代码。
  - 用户主目录路径可通过`Sys.getenv("HOME")`获取或者刚打开R时通过`getwd()`获得。
  - 在R中执行`file.edit('~/.Rprofile')`，然后将相关代码(见下页)写入并保存。
- **Rstudio配置**: (菜单`Tools -> Global Options`)
  - 取消自动加载.RData到工作环境中: `General -> Restore .RData into workspace at startup`
  - 代码文件格式保存为UTF-8: `Code -> Saving -> Default text encoding -> UTF-8`
  - 代码自动换行: `Code -> Editing -> Soft-wrap R source files`

[*] RobertI.Kabacoff, 卡巴科弗, 陈钢,等. R语言实战. 2013. 附录B.

```r
# .Rprofile
# 设置常用选项
options(papersize="a4")
options(editor="notepad")
options(tab.width=2)
options(width=130)
options(digits=4)
options(stringsAsFactors=FALSE)
grDevices::windows.options(record=TRUE)
# 设置R交互提示信息
options(prompt="> ")
options(continue="+ ")

# 设置包的本地库（library）路径
.libPaths("~/Library_R") # 需在用户主目录下新建文件夹 Library_R
# 设置之后重启R，.libPaths()返回新建的本地库路径

# 设置CRAN镜像默认地址
options(repos = c(CRAN = "http://cran.r-project.org/",
                  CRANextra = "https://cloud.r-project.org/"))

# 启动函数
.First = function(){ cat("\nWelcome at", base::date(), "\n") }

# 会话结束函数
.Last = function(){ cat("\nGoodbye at ", base::date(), "\n") }
```

R包与相关资料

以scorecard包(Github, Slide)为例:

- 安装包,

```
# install from CRAN
install.packages("scorecard")
# install from github
devtools::install_github(
  "shichenxie/scorecard")
```

- 加载包, library(scorecard)
- 获取帮助:
  - 包的帮助 help(package="scorecard")
  - 函数的帮助 help(woebin)或者?woebin

R包与相关资料

以scorecard包(Github, Slide)为例:

- 安装包,

```
# install from CRAN
install.packages("scorecard")
# install from github
devtools::install_github(
  "shichenxie/scorecard")
```

- 加载包, `library(scorecard)`
- 获取帮助:
  - 包的帮助 `help(package="scorecard")`
  - 函数的帮助 `help(woebin)`或者`?woebin`

- 目前(2017.11)CRAN - Packages上有11815个包，如何找到有用的R包:
  - CRAN Task Views, CRAN将部分优质包分主题归纳，目前有35个Views
  - R语言学习路径
- R语言相关网站:
  - R官网
  - Rstudio
  - r-bloggers
  - Quick-R
  - 统计之都COS
  - 大神: 谢益辉(HP, GH), Hadley(HP, GH), Matt(GH)

3. scorecard包的使用

评分卡开发流程*

1. 数据准备
    - 特征衍生与数据整合
    - WOE转换与单变量统计

评分卡开发流程*

1. 数据准备
    - 特征衍生与数据整合
    - WOE转换与单变量统计
2. 变量筛选
    - 变量粗筛: 信息值、缺失率、单类别比例
    - 模型筛选: LASSO、step

评分卡开发流程*

1. 数据准备
    - 特征衍生与数据整合
    - WOE转换与单变量统计
2. 变量筛选
    - 变量粗筛: 信息值、缺失率、单类别比例
    - 模型筛选: LASSO、step
3. 模型开发
    - 逻辑回归模型

评分卡开发流程*

1. 数据准备
    - 特征衍生与数据整合
    - WOE转换与单变量统计
2. 变量筛选
    - 变量粗筛: 信息值、缺失率、单类别比例
    - 模型筛选: LASSO、step
3. 模型开发
    - 逻辑回归模型
4. 模型验证
    - 预测模型需要满足：准确性、稳健性、有意义

评分卡开发流程*

1. 数据准备
    - 特征衍生与数据整合
    - WOE转换与单变量统计
2. 变量筛选
    - 变量粗筛: 信息值、缺失率、单类别比例
    - 模型筛选: LASSO、step
3. 模型开发
    - 逻辑回归模型
4. 模型验证
    - 预测模型需要满足：准确性、稳健性、有意义
5. 评分卡刻度与实施

评分卡开发流程*

1. 数据准备
    - 特征衍生与数据整合
    - WOE转换与单变量统计
2. 变量筛选
    - 变量粗筛: 信息值、缺失率、单类别比例
    - 模型筛选: LASSO、step
3. 模型开发
    - 逻辑回归模型
4. 模型验证
    - 预测模型需要满足：准确性、稳健性、有意义
5. 评分卡刻度与实施

[*] 马姆杜·雷法特, 王松奇, 林治乾. 信用风险评分卡研究[M]. 社会科学文献出版社, 2013.

1. 数据准备-样本切分

通常将样本分为训练集与测试集，通过评估模型在两个数据集上的表现，并进而选择模型。
scorecard::split_df函数可随机将样本分层切分为训练集与测试集:

```
library(data.table)
library(scorecard)
# 加载数据
data("germancredit")

# y变量creditability赋值为0/1
dt = setDT(germancredit)[
  , creditability := ifelse(creditability=="bad",1,0)]

# 数据集切分为训练集与测试集
dt_list = split_df(dt, y="creditability", ratio=0.6, seed=21)
train = dt_list$train; test = dt_list$test
```

WOE转换

```
# 分箱
bins = woebin(dt, y="creditability", print_step=0)

class(bins)

# [1] "list"

bins$age.in.years

#        variable       bin count count_distr good bad badprob     woe
# 1: age.in.years [-Inf,26)   190       0.190  110  80  0.4211  0.5288
# 2: age.in.years   [26,28)   101       0.101   74  27  0.2673 -0.1609
# 3: age.in.years   [28,35)   257       0.257  172  85  0.3307  0.1425
# 4: age.in.years   [35,37)    79       0.079   67  12  0.1519 -0.8725
# 5: age.in.years [37, Inf)   373       0.373  277  96  0.2574 -0.2124
#      bin_iv total_iv
# 1: 0.057921   0.1305
# 2: 0.002529   0.1305
# 3: 0.005359   0.1305
# 4: 0.048610   0.1305
# 5: 0.016080   0.1305
```

- 分箱i的WOE值(Weight of Evidence)定义如下:

$$WOE_i = \ln\left[\frac{Bad\_Distr_i}{Good\_Distr_i}\right] = \ln\left[\frac{Bad_i/\sum Bad_i}{Good_i/\sum Good_i}\right] = \ln\left[\frac{Bad_i/Good_i}{\sum Bad_i/\sum Good_i}\right]$$

即，WOE值是分箱i的坏客户分布与好客户分布的比值的对数，调整为分箱i的坏好比与总体样本的坏好比的比值的对数，衡量了分箱i对整体坏好比的影响程度。

- 分箱i的WOE值(Weight of Evidence)定义如下:

$$WOE_i = \ln\left[\frac{Bad\_Distr_i}{Good\_Distr_i}\right] = \ln\left[\frac{Bad_i / \sum Bad_i}{Good_i / \sum Good_i}\right] = \ln\left[\frac{Bad_i / Good_i}{\sum Bad_i / \sum Good_i}\right]$$

  即，WOE值是分箱i的坏客户分布与好客户分布的比值的对数，调整为分箱i的坏好比与总体样本的坏好比的比值的对数，衡量了分箱i对整体坏好比的影响程度。

- 信息值(Information Value, IV)是衡量一个二元变量y和一个名义变量x之间的关联性的指标。信息值定义如下:

$$IV = \sum_i (Bad\_Distr_i - Good\_Distr_i) \ln\left(\frac{Bad\_Distr_i}{Good\_Distr_i}\right)$$

- 分箱i的WOE值(Weight of Evidence)定义如下:

$$WOE_i = \ln\left[\frac{Bad\_Distr_i}{Good\_Distr_i}\right] = \ln\left[\frac{Bad_i/\sum Bad_i}{Good_i/\sum Good_i}\right] = \ln\left[\frac{Bad_i/Good_i}{\sum Bad_i/\sum Good_i}\right]$$

  即，WOE值是分箱i的坏客户分布与好客户分布的比值的对数，调整为分箱i的坏好比与总体样本的坏好比的比值的对数，衡量了分箱i对整体坏好比的影响程度。

- 信息值(Information Value, IV)是衡量一个二元变量y和一个名义变量x之间的关联性的指标。信息值定义如下:

$$IV = \sum_i (Bad\_Distr_i - Good\_Distr_i) \ln\left(\frac{Bad\_Distr_i}{Good\_Distr_i}\right)$$

```
# 信息值函数 scorecard::iv
ivs = iv(germancredit,
         y="creditability")

ivs[variable=="age.in.years",]

#         variable info_value
# 1: age.in.years     0.2597
```

- 分箱i的WOE值(Weight of Evidence)定义如下:

$$WOE_i = \ln\left[\frac{Bad\_Distr_i}{Good\_Distr_i}\right] = \ln\left[\frac{Bad_i/\sum Bad_i}{Good_i/\sum Good_i}\right] = \ln\left[\frac{Bad_i/Good_i}{\sum Bad_i/\sum Good_i}\right]$$

即，WOE值是分箱i的坏客户分布与好客户分布的比值的对数，调整为分箱i的坏好比与总体样本的坏好比的比值的对数，衡量了分箱i对整体坏好比的影响程度。

- 信息值(Information Value, IV)是衡量一个二元变量y和一个名义变量x之间的关联性的指标。信息值定义如下:

$$IV = \sum_i (Bad\_Distr_i - Good\_Distr_i)\ln\left(\frac{Bad\_Distr_i}{Good\_Distr_i}\right)$$

```
# 信息值函数 scorecard::iv
ivs = iv(germancredit,
         y="creditability")

ivs[variable=="age.in.years",]

#         variable info_value
# 1: age.in.years     0.2597
```

通过信息值大小判断候选自变量的预测力

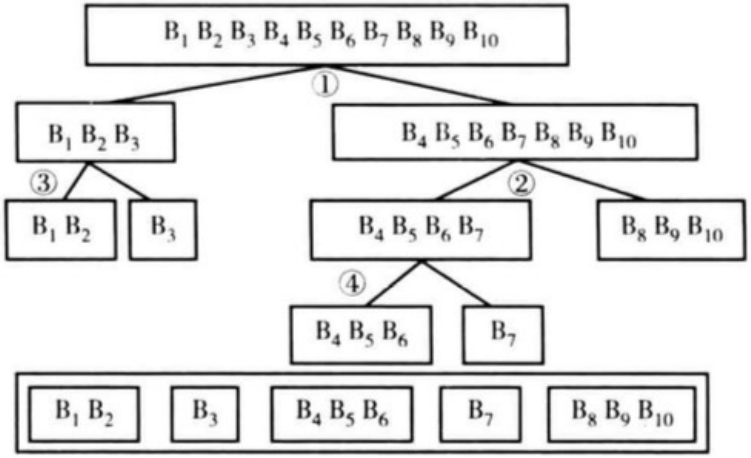
IV范围	预测力
小于0.02	无预测力
0.02到0.10	弱
0.10到0.30	中等
大于0.30	强

- scorecard::woebin分箱函数：模型候选变量包括连续变量、类别变量、有序类别变量三类，分别对应R语言中的numeric、character、factor数据类型(logical变量分箱时与numeric类似)。
    - 连续变量首先需等距分段，woebin函数通过min_perc_total控制，默认值为0.02，即初始等距分50段。且极大/小值*单独分段。
    - 对于类别变量，根据每个类别变量的坏客户率排序。
    - 通过上面两步连续变量与类别变量均转换为有序类别变量。

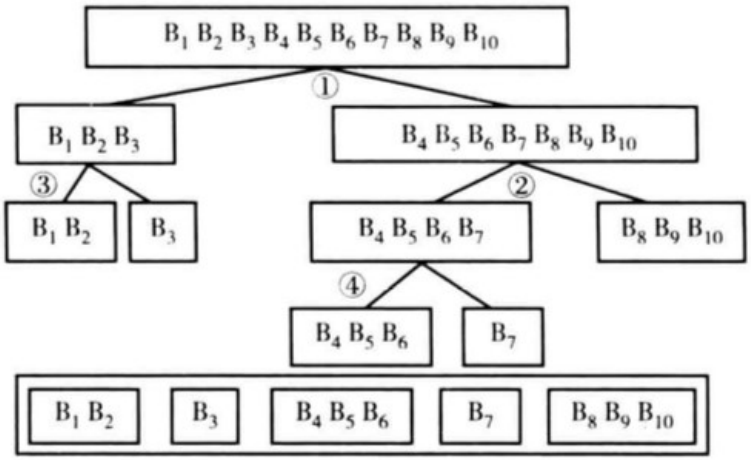
- scorecard::woebin分箱函数：模型候选变量包括连续变量、类别变量、有序类别变量三类，分别对应R语言中的numeric、character、factor数据类型(logical变量分箱时与numeric类似)。
    - 连续变量首先需等距分段，woebin函数通过min_perc_total控制，默认值为0.02，即初始等距分50段。且极大/小值*单独分段。
    - 对于类别变量，根据每个类别变量的坏客户率排序。
    - 通过上面两步连续变量与类别变量均转换为有序类别变量。
    - 最后，通过决策树实现有序类别变量的最优分箱，其算法过程见下一页
        - 最优分箱过程中需确保分箱区间的有序性
        - 每一个最优分箱节点使得分箱之后的信息值在所有候选分箱节点中最大
        - 分箱停止条件是信息增益率stop_limit与最大分箱数 max_bin_num，默认值分别为0.1与6，即信息增益率小于10%或者非缺失值分箱数大于6后停止进一步分箱。

[*] 此处极端值定义为：均值上下3倍四分位距以外的值，其中四分位距指上下四分位数之差。

决策树最优分箱算法过程：



决策树最优分箱算法过程：

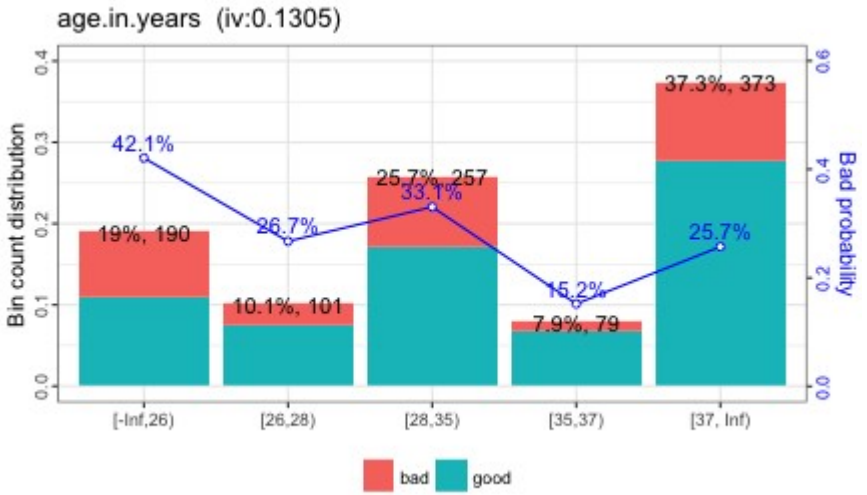


`scorecard::woebin_plot`绘制分箱信息图。合理的分箱应有以下特征：

- 坏客户率或者WOE值趋势线，最多不超过一个拐点，最好是单调的
- 每个分箱的样本数量占比最好大于5%

```
woebin_plot(bins$age.in.years)
```

```
## $age.in.years
```

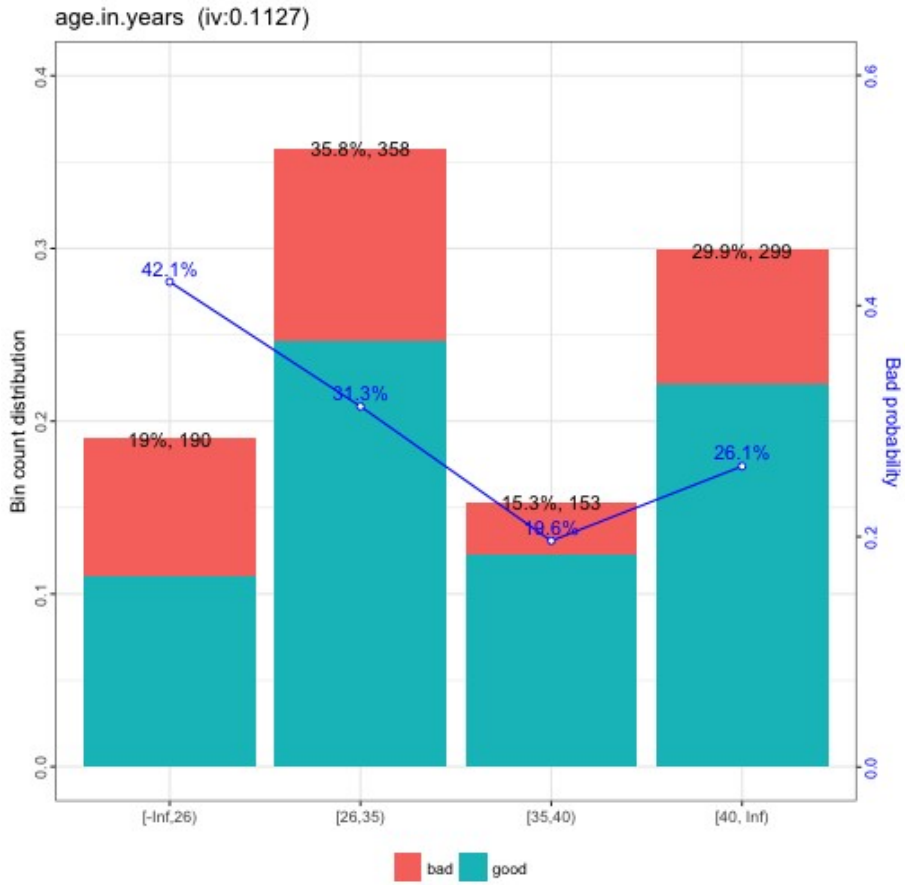


```
# 手动调整分箱
break_adj = list(
  age.in.years=c(26,35,40))

bins_adj = woebin(
  dt, y="creditability",
  breaks_list=break_adj,
  print_step=0)

woebin_plot(bins_adj$age.in.years)
```

```
# $age.in.years
```




```
# 调整后分箱信息
bins_adj$age.in.years[,.(bin,woe)]
```

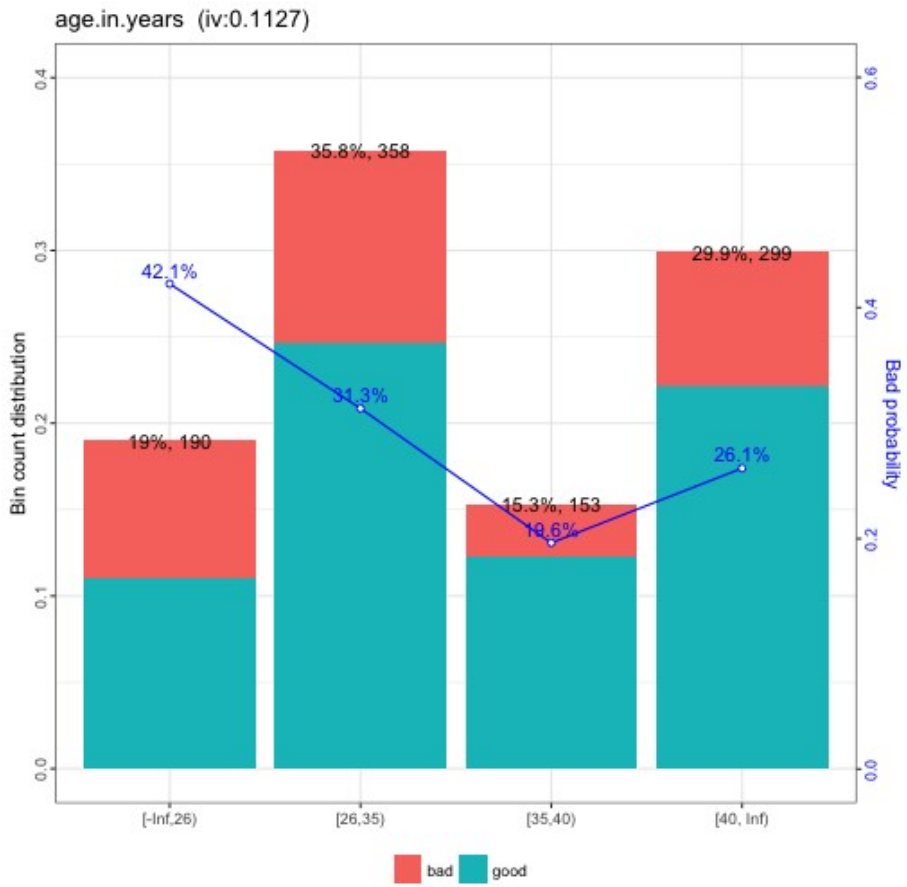
```
##           bin      woe
## 1: [-Inf,26)  0.52884
## 2:   [26,35)  0.06047
## 3:   [35,40) -0.56369
## 4: [40, Inf) -0.19416
```

```
# 手动调整分箱
break_adj = list(
  age.in.years=c(26,35,40))

bins_adj = woebin(
  dt, y="creditability",
  breaks_list=break_adj,
  print_step=0)

woebin_plot(bins_adj$age.in.years)
```

```
# $age.in.years
```




```
# 调整后分箱信息
bins_adj$age.in.years[,.(bin,woe)]
```

```
##           bin      woe
## 1: [-Inf,26)  0.52884
## 2:   [26,35)  0.06047
## 3:   [35,40) -0.56369
## 4: [40, Inf) -0.19416
```

```
# WOE转换
train_woe = woebin_ply(
  train, bins_adj, print_step=0)
test_woe = woebin_ply(
  test, bins_adj, print_step=0)

cbind(train[1:5,.(age.in.years)],
      train_woe[1:5,.(age.in.years_woe)])
```

```
#    age.in.years age.in.years_woe
# 1:           34          0.06047
# 2:           28          0.06047
# 3:           31          0.06047
# 4:           46         -0.19416
# 5:           28          0.06047
```

2. 变量选择-变量粗筛

scorecard::var_filter提供了变量粗筛功能，默认删除信息值<0.02、缺失率>95%、单类别比例>95%的变量

```
dt_sel=var_filter(train_woe,y="creditability") #粗筛后数据集
var_sel = names(dt_sel)
```

2. 变量选择-变量粗筛

scorecard::var_filter提供了变量粗筛功能，默认删除信息值<0.02、缺失率>95%、单类别比例>95%的变量

```
dt_sel=var_filter(train_woe,y="creditability") #粗筛后数据集
var_sel = names(dt_sel)
```

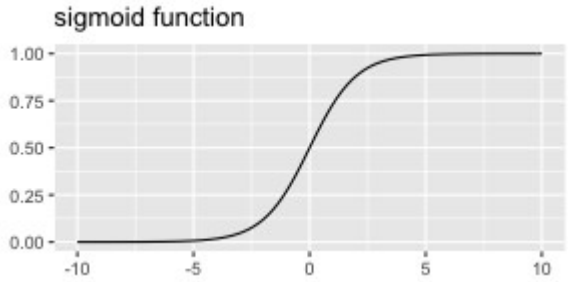
```
dt_sel2=var_filter(train_woe,y="creditability",return_rm_reason=TRUE)
# dt_sel2$dt # 返回粗筛后数据集 # dt_sel2$rm # 返回删除变量的原因
```

variable	rm_reason
installment.rate.in.percentage.of.disposable.income_woe	iv < 0.02
present.residence.since_woe	iv < 0.02
job_woe	iv < 0.02
number.of.people.being.liable.to.provide.maintenance.for_woe	iv < 0.02
telephone_woe	iv < 0.02
foreign.worker_woe	identical rate > 0.95

3. 模型开发-逻辑回归

逻辑回归(logistic regression)在信用评分卡开发中起到核心作用。逻辑回归通过sigmoid函数 $y = 1/(1+e^{-z})$ 将线性回归模型 $z = \boldsymbol{w}^T\boldsymbol{x} + b$ 产生的预测值转换为一个接近0或1的拟合值 $\hat{y}$ ：

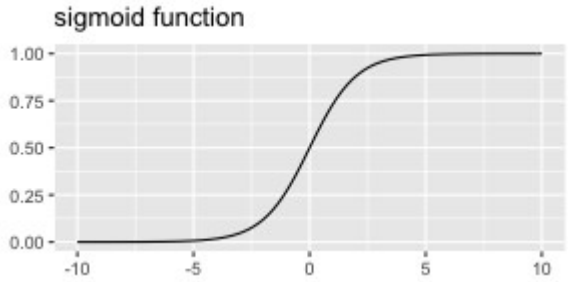
$$\begin{aligned}\hat{y} &= \frac{1}{1+e^{-z}} \\ &= \frac{1}{1+e^{-(\boldsymbol{w}^T\boldsymbol{x}+b)}}\end{aligned}$$



3. 模型开发-逻辑回归

逻辑回归(logistic regression)在信用评分卡开发中起到核心作用。逻辑回归通过sigmoid函数 $y=1/(1+e^{-z})$ 将线性回归模型 $z=\boldsymbol{w}^T\boldsymbol{x}+b$ 产生的预测值转换为一个接近0或1的拟合值 $\hat{y}$ ：

$$\begin{aligned}\hat{y} &= \frac{1}{1+e^{-z}} \\ &= \frac{1}{1+e^{-(\boldsymbol{w}^T\boldsymbol{x}+b)}}\end{aligned}$$



上式的 $\hat{y}$ 可视为事件发生的概率 $p(y=1|\boldsymbol{x})$ ，变换后得到：

$$\ln\frac{p}{1-p}=z=\boldsymbol{w}^T\boldsymbol{x}+b$$

其中， $p/(1-p)$ 为比率(odds)，即违约概率与正常概率的比值。 $\ln p/(1-p)$ 为logit函数，即比率的自然对数。因此，逻辑回归实际上是用比率的自然对数作为因变量的线性回归模型。

逻辑回归代价函数(cost function)

单个样本的损失函数(loss function):

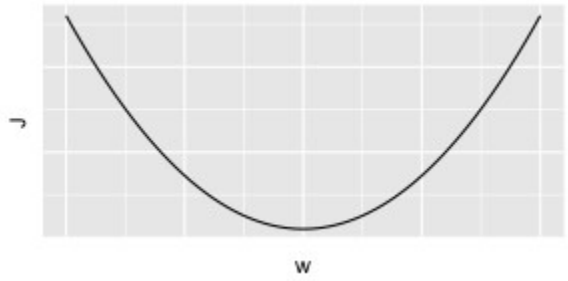
$$\ell(\hat{y}, y) = -(y \ln \hat{y} + (1-y) \ln (1-\hat{y}))$$

对于整个训练集的代价函数(cost function):

$$\begin{aligned} J(\boldsymbol{w}, b) &= \frac{1}{m} \sum_i \ell(\hat{\boldsymbol{y}}^{(i)}, \boldsymbol{y}^{(i)}) \\ &= -\frac{1}{m} \sum_i [\boldsymbol{y}^{(i)} \ln \hat{\boldsymbol{y}}^{(i)} + (1-\boldsymbol{y}^{(i)}) \ln (1-\hat{\boldsymbol{y}}^{(i)})] \end{aligned}$$

其中, $\hat{y}$ 为拟合值, y 为实际标签, m 为样本数量

使用梯度下降(gradient descent), 找到合适的参数 $(\boldsymbol{w}, b)$, 使得 $J(\boldsymbol{w}, b)$ 尽可能小:



$$\begin{aligned} &\min J(\boldsymbol{w}, b): \\ &repeat\{ \\ &\boldsymbol{w} := \boldsymbol{w} - \alpha \frac{dJ}{dw} \\ &\} \end{aligned}$$

正则化(regulation)

在使得代价函数最小时，尤其当样本特征很多时，容易陷入过拟合问题。为了改善过拟合，通常在代价函数中引入正则化项：

$$\min J(\boldsymbol{w}, b) + \frac{\lambda}{m}(\alpha||\boldsymbol{w}||_1 + \frac{1}{2}(1-\alpha)||\boldsymbol{w}||_2^2)$$

其中，正则化参数 $\lambda > 0$ ；$||\boldsymbol{w}||_1 = \sum_j |w_j|$ 与 $||\boldsymbol{w}||_2^2 = \sum_j w_j^2 = \boldsymbol{w}^T\boldsymbol{w}$ 分别为L1与L2范数正则化，也分别称为LASSO(Least Absolute Shrinkage and Selection Operator)与"岭回归"(ridge regression)。L1与L2范数正则化都有助于降低过拟合风险，但前者更易于获得稀疏解，即求得的 $\boldsymbol{w}$ 会有更少的非零分量。使用lasso筛选变量的案例如下：

```
library(h2o)
localH2O = h2o.init()
dth2o = as.h2o(train_woe)
# h2o.glm lasso
fit = h2o.glm(y="creditability", training_frame=dth2o,
family="binomial", nfolds=0, alpha=1, lambda_search=TRUE) # summary(fit)

# variable importance
varimp = data.table(h2o.varimp(fit))[names!=""][!is.na(coefficients) & coefficients > 0
var_sel3 = c(varimp$names, "creditability")
```

基于AIC筛选变量step

```
# glm ------
m1 = glm( creditability ~ ., family = "binomial", data = train_woe[,var_sel,with=FALSE])
# summary(m1)

# Select a formula-based model by AIC
m_step = step(m1, direction="both", trace = FALSE)
m2 = eval(m_step$call)
summary(m2)$coefficients
```

```
##                                         Estimate Std. Error z value
## (Intercept)                              -0.7806     0.1038  -7.522
## status.of.existing.checking.account_woe   0.5764     0.1305   4.418
## duration.in.month_woe                     0.6085     0.2048   2.972
## credit.history_woe                        0.4244     0.1973   2.151
## purpose_woe                               1.0134     0.2724   3.721
## credit.amount_woe                         0.6753     0.2370   2.850
## savings.account.and.bonds_woe             0.6752     0.2398   2.815
## present.employment.since_woe              0.5946     0.3595   1.654
## personal.status.and.sex_woe               1.1846     0.5174   2.290
## property_woe                              0.7705     0.3264   2.361
## age.in.years_woe                          0.7617     0.3210   2.373
## other.installment.plans_woe               0.7168     0.4050   1.770
##                                          Pr(>|z|)
## (Intercept)                             5.389e-14
## status.of.existing.checking.account_woe 9.956e-06
## duration.in.month_woe                   2.958e-03
## credit.history_woe                      3.146e-02
## purpose_woe                             1.988e-04
## credit.amount_woe                       4.371e-03
## savings.account.and.bonds_woe           4.874e-03
## present.employment.since_woe            9.810e-02
## personal.status.and.sex_woe             2.205e-02
## property_woe                            1.825e-02
## age.in.years_woe                        1.767e-02
## other.installment.plans_woe             7.674e-02
```

4. 模型验证评估

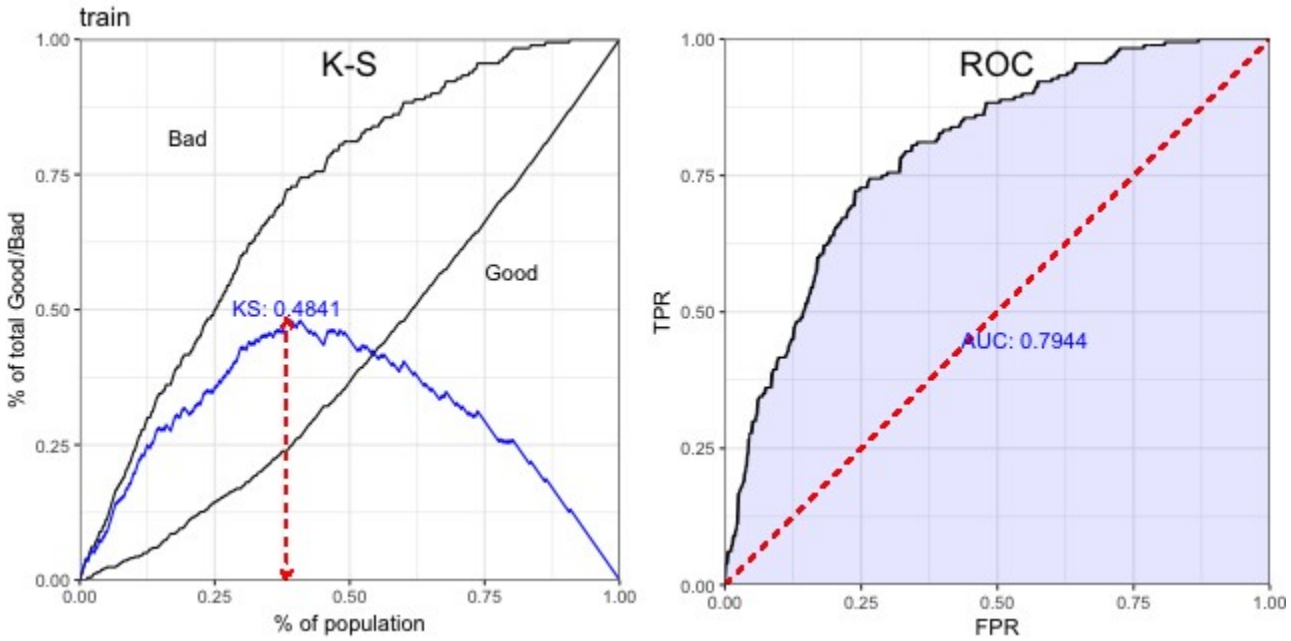
评分卡要求备选的逻辑回归模型达到三个基本要求:

- 精确性，达到可接受水平。
- 稳健性，要求能够适用于更广范围的数据集。
- 有意义，即业务变量及其预测值是可解释的。例如，信用卡的额度利用率越高，违约率相应也越高。

scorecard::perf_eva函数使用预测违约概率与实际违约标签，给出了模型评估的主要指标(KS值、AUC、Gini)与相应图表(KS曲线、ROC曲线、提升图、PR曲线)。

```
# predicted proability
train_pred = predict(m2, type='response', train_woe)
test_pred = predict(m2, type='response', test_woe)

# performance
perf_train = perf_eva(train$creditability, train_pred, title="train")
```




```
perf_test = perf_eva(test$creditability, test_pred, title="test")
```

KS

- KS(柯尔莫哥洛夫-斯米尔诺夫kolmogorov-smirnow)图纵轴为坏客户累计百分比，横轴为总体样本累计百分比。perf_eva函数绘制KS曲线过程:
  - 先将样本随机排列，随机种子seed默认为186
  - 按照预测违约概率倒序排列(坏客户累计百分比曲线位于上方)
  - 分为groupnum(默认20)等份
  - 计算每一等份中违约与正常客户的累计百分比
  - 绘制出两者之间差值即为KS曲线
- KS曲线中的最大值即为KS值，其取值范围0~1。KS值越大模型的区分能力越好。
- 通常申请评分卡要求$KS \geq 0.3$。而且测试集与训练集的KS值相差小于0.01。

KS

- KS(柯尔莫哥洛夫-斯米尔诺夫kolmogorov-smirnow)图纵轴为坏客户累计百分比，横轴为总体样本累计百分比。perf_eva函数绘制KS曲线过程:
  - 先将样本随机排列，随机种子seed默认为186
  - 按照预测违约概率倒序排列(坏客户累计百分比曲线位于上方)
  - 分为groupnum(默认20)等份
  - 计算每一等份中违约与正常客户的累计百分比
  - 绘制出两者之间差值即为KS曲线
- KS曲线中的最大值即为KS值，其取值范围0~1。KS值越大模型的区分能力越好。
- 通常申请评分卡要求KS $\geq$ 0.3。而且测试集与训练集的KS值相差小于0.01。

ROC与AUC

- ROC(受试者工作特征曲线Receiver Operating Characteristic)曲线纵轴为真正例率(True Positive Rate, TPR)，横轴为假正例率(False Positive Rate, FPR)：
  - 先将样本随机排列，随机种子seed默认为186
  - 按照预测违约概率降序排列
  - 分概率值计算好坏客户数量，然后计算TPR=TP/(TP+FN)与FPR=FP/(TN+FP)
  - 以TPR为纵轴FPR为横轴绘制散点图即为ROC曲线
- AUC(Area Under ROC Curve)为ROC曲线下面积之和，其取值范围0~1。AUC值越大模型效果越好。
- 行为评分卡通常要求AUC $\geq$ 0.75，申请评分卡的AUC相对低一些也能够接受。

模型的整体评估

一个好的模型一般应具有以下特征：

模型的整体评估

一个好的模型一般应具有以下特征：

1. 在进行数据描述时变量应该有意义。通常，某些变在特定客群的不同风险模型中重复出现。例如，信用卡行为评分卡模型中，授信使用率经常出现；申请评分卡模型中收入水平、职业和历史信贷产品拥有情况比人口统计变量重要。

模型的整体评估

一个好的模型一般应具有以下特征：

1. 在进行数据描述时变量应该有意义。通常，某些变在特定客群的不同风险模型中重复出现。例如，信用卡行为评分卡模型中，授信使用率经常出现；申请评分卡模型中收入水平、职业和历史信贷产品拥有情况比人口统计变量重要。

2. 变量的预测力或贡献度，应该在模型的变量之间分布。

模型的整体评估

一个好的模型一般应具有以下特征：

1. 在进行数据描述时变量应该有意义。通常，某些变在特定客群的不同风险模型中重复出现。例如，信用卡行为评分卡模型中，授信使用率经常出现；申请评分卡模型中收入水平、职业和历史信贷产品拥有情况比人口统计变量重要。

2. 变量的预测力或贡献度，应该在模型的变量之间分布。

3. 模型中不应该包含太多变量。通常，包含的变量不超过9~20个(最优10~12个)。变量太多可能导致过拟合，变量太少往往区分度不够。

模型的整体评估

一个好的模型一般应具有以下特征：

1. 在进行数据描述时变量应该有意义。通常，某些变在特定客群的不同风险模型中重复出现。例如，信用卡行为评分卡模型中，授信使用率经常出现；申请评分卡模型中收入水平、职业和历史信贷产品拥有情况比人口统计变量重要。

2. 变量的预测力或贡献度，应该在模型的变量之间分布。

3. 模型中不应该包含太多变量。通常，包含的变量不超过9~20个(最优10~12个)。变量太多可能导致过拟合，变量太少往往区分度不够。

4. 最终模型的变量应该能够确保包含稳健一致的数据，并在后续实施阶段能够准确获取。

5. 评分卡刻度与实施

- 评分卡的分值刻度通过将分值表示为比率对数的线性表达式:

$$score = A - B\ln(odds)$$

其中， A与B是常数, 坏好比率 $odds = p/(1-p)$ 为一个客户违约的估计概率与正常的估计概率的比率, $\ln(odds)$ 为逻辑回归的因变量，即 $\ln(odds) = \boldsymbol{w}^T\boldsymbol{x} + b$

5. 评分卡刻度与实施

- 评分卡的分值刻度通过将分值表示为比率对数的线性表达式:

$$score = A - B\ln(odds)$$

其中， A与B是常数, 坏好比率 $odds = p/(1-p)$ 为一个客户违约的估计概率与正常的估计概率的比率, $\ln(odds)$ 为逻辑回归的因变量，即 $\ln(odds) = \boldsymbol{w}^T\boldsymbol{x} + b$

- 常数A和B的值可以通过两个假设代入上式计算得到：
  - 基准坏好比率(odds0)对应的基准分值(points0)
    - $points0 = A - B\ln(odds0)$
  - 坏好比率翻倍的分数PDO(Points to Double the Odds)
    - $points0 - PDO = A - B\ln(2odds0)$
  - 解上述两方程，可以得到：
    - $B = PDO/\ln(2)$
    - $A = points0 + B\ln(odds0)$

评分卡刻度

- 分值分配。将逻辑回归公式代入评分卡分值公式，可以得到

$$\begin{aligned} score &= A - B\ln(odds) = A - B(\boldsymbol{w}^T\boldsymbol{x} + b) \\ &= (A - Bb) - Bw_1x_1 - Bw_2x_2 \cdots - Bw_mx_m \end{aligned}$$

  其中， $x_1 \cdots x_m$ 为最终进入模型的自变量且已经转换为WOE值, w_i 为逻辑回归的变量系数, b 为逻辑回归的截距, A, B 为上页求得的刻度因子。 Bw_ix_i 为变量 x_i 对应的评分， $(A - Bb)$ 为基础分(也可将基础分值平均分配给各个变量)。

评分卡刻度

- 分值分配。将逻辑回归公式代入评分卡分值公式，可以得到

$$\begin{aligned} score &= A - B\ln(odds) = A - B(\boldsymbol{w}^T\boldsymbol{x} + b) \\ &= (A - Bb) - Bw_1x_1 - Bw_2x_2 \cdots - Bw_mx_m \end{aligned}$$

  其中， $x_1 \cdots x_m$ 为最终进入模型的自变量且已经转换为WOE值, w_i 为逻辑回归的变量系数, b 为逻辑回归的截距, A, B 为上页求得的刻度因子。 Bw_ix_i 为变量 x_i 对应的评分， $(A - Bb)$ 为基础分(也可将基础分值平均分配给各个变量)。

- `scorecard::scorecard`函数创建评分卡刻度，其默认设置`points0=600, odds0=1/19, pdo=50`

```
# 创建评分卡刻度
card = scorecard(bins_adj, m2)
data.table::rbindlist(card, fill=TRUE)[c(1,37:40),1:4]
```

```
##        variable       bin      woe points
## 1:   basepoints        NA       NA    444
## 2: age.in.years [-Inf,26)  0.52884    -29
## 3: age.in.years   [26,35)  0.06047     -3
## 4: age.in.years   [35,40) -0.56369     31
## 5: age.in.years [40, Inf) -0.19416     11
```

评分卡刻度(续)

- scorecard::scorecard_ply函数可将原始数据集转换为信用评分

```
# 评分转换只有总分
train_score = scorecard_ply(train, card, only_total_score=TRUE, print_step=0)
test_score = scorecard_ply(test, card, only_total_score=TRUE, print_step=0)
str(train_score)
```

```
## Classes 'data.table' and 'data.frame':    600 obs. of  1 variable:
##  $ score: num  402 423 420 628 389 361 373 699 401 423 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# 评分转换包括变量分值
train_score2 = scorecard_ply(train, card, only_total_score = FALSE, print_step=0)
str(train_score2)
```

```
## Classes 'data.table' and 'data.frame':    600 obs. of  12 variables:
##  $ status.of.existing.checking.account_points: num  -26 -26 -26 49 17 -26 -26 49 -26 49 ...
##  $ duration.in.month_points                  : num  15 -5 58 -5 -5 -5 -23 58 15 15 ...
##  $ credit.history_points                     : num  -3 -3 -3 22 -3 -35 -3 22 -3 -3 ...
##  $ purpose_points                            : num  -20 -20 -20 30 -20 59 -20 30 -20 -20 ...
##  $ credit.amount_points                      : num  17 -14 17 -5 17 17 17 17 -5 17 ...
##  $ savings.account.and.bonds_points          : num  -13 -13 -7 37 -7 -13 -13 37 -13 -13 ...
##  $ present.employment.since_points           : num  -18 17 10 -1 -1 -1 -18 -1 -1 -18 ...
##  $ personal.status.and.sex_points            : num  -23 14 -23 14 -23 -23 14 14 12 -23 ...
##  $ property_points                           : num  26 26 -2 26 -2 -2 -2 26 26 -2 ...
##  $ age.in.years_points                       : num  -3 -3 -3 11 -3 -29 -3 -3 -3 -29 ...
##  $ other.installment.plans_points            : num  6 6 -25 6 -25 -25 6 6 -25 6 ...
##  $ score                                     : num  402 423 420 628 389 361 373 699 401 423 .
##  - attr(*, ".internal.selfref")=<externalptr>
```

评分稳定性指数

稳定性指数(population stability index, PSI)是计算实际和预期的分值分布之间差异的一个衡量指标, $PSI = \sum_i (A_i - E_i) \ln(A_i/E_i)$ 。其计算过程见下表:

bin	A	E	A_E	logA_E	PSI	total_PSI
[150,300)	0.0333	0.0425	-0.0092	-0.2429	0.0022	0.0266
[300,350)	0.0683	0.0725	-0.0042	-0.0592	0.0002	0.0266
[350,400)	0.1517	0.1625	-0.0108	-0.0690	0.0007	0.0266
[400,450)	0.1767	0.1700	0.0067	0.0385	0.0003	0.0266
[450,500)	0.1900	0.2175	-0.0275	-0.1352	0.0037	0.0266
[500,550)	0.1683	0.1650	0.0033	0.0200	0.0001	0.0266
[550,600)	0.1233	0.0825	0.0408	0.4021	0.0164	0.0266
[600,650)	0.0667	0.0725	-0.0058	-0.0839	0.0005	0.0266
[650,700)	0.0217	0.0150	0.0067	0.3677	0.0025	0.0266

- 稳定性指数PSI与信息值的计算公式相同。信息值衡量的是两个离散变量之间的关联性，较低的取值表明两个变量的类别分布相似。信用评分卡使用稳定性指数遵循的准则如下：

指数范围	解释
0~0.1	无显著变化，无须采取行动
0.1~0.25	发现某些变化，建议进行检查
>0.25	发现显著变化，建议重新建立评分卡

- 稳定性指数PSI与信息值的计算公式相同。信息值衡量的是两个离散变量之间的关联性，较低的取值表明两个变量的类别分布相似。信用评分卡使用稳定性指数遵循的准则如下：

指数范围	解释
0~0.1	无显著变化，无须采取行动
0.1~0.25	发现某些变化，建议进行检查
>0.25	发现显著变化，建议重新建立评分卡

- 稳定性指数可以用于以下三个目的：
    - 作为验证统计量，以确保训练数据集与测试数据集得到的评分分布之间没有显著差异。
    - 作为监控评分卡实施以后表现的控制措施。如果稳定性指数表明发生显著变化，需要调查原因，必要时甚至需要重建评分卡。
    - 还可以监测预测变量的评分分布是否发生变化。

scorecard::perf_psi函数计算两个样本的评分稳定性指数并绘制评分分布图。如果参数score输入多个评分或多个变量的评分，能够同时返回相应的稳定性指数。

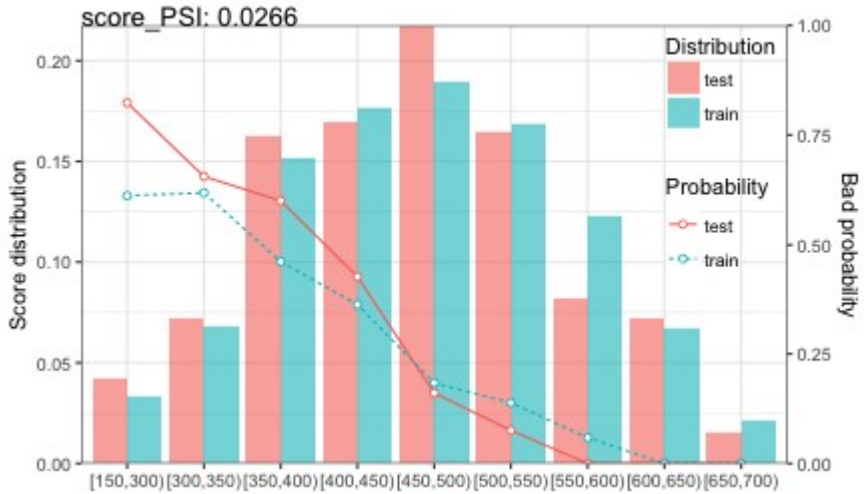
```
psi = perf_psi(
  score = list(
    train=train_score,
    test =test_score),
  label = list(
    train=train$creditability,
    test =test$creditability),
  x_limits = c(250, 750),
  x_tick_break = 50)

psi$psi
```

```
##    variable    PSI
## 1:    score 0.0266
```

```
psi$pic
```

```
## $score
```



评分卡建模案例

```
library(data.table)
library(scorecard)
# load germancredit data
data("germancredit")

# rename creditability as y
dt = setDT(germancredit)[, `:=`(
  y = ifelse(creditability == "bad", 1, 0),
  creditability = NULL)]

# filter variable via iv, missing rate, identical value rate
dt_s = var_filter(dt, "y")

# breaking dt into train and test ------
dt_list = split_df(dt_s, y="y", ratio = 0.6, seed = 30)
train = dt_list$train; test = dt_list$test;

# woe binning ------
bins = woebin(train, "y", print_step=0) # woebin_plot(bins)
# binning adjustment
breaks_adj = list(age.in.years=c(26, 35, 40))
bins_adj = woebin(dt_s,"y", breaks_list=breaks_adj, print_step=0)

# converting train and test into woe values
train_woe = woebin_ply(train, bins_adj, print_step=0)
test_woe = woebin_ply(test, bins_adj, print_step=0)
```

```r
# glm ------
m1 = glm( y ~ ., family = "binomial", data = train_woe) # summary(m1)

# Select a formula-based model by AIC
m_step = step(m1, direction="both", trace = FALSE)
m2 = eval(m_step$call) # summary(m2)

# performance ------
# predicted proability
train_pred = predict(m2, train_woe, type='response')
test_pred = predict(m2, test_woe, type='response')

# ks & roc plot
perf_eva(train$y, train_pred, title = "train")
perf_eva(test$y, test_pred, title = "test")

# score
card = scorecard(bins_adj, m2)
# credit score, only_total_score = TRUE
train_score = scorecard_ply(train, card, only_total_score=TRUE, print_step=0)
test_score = scorecard_ply(test, card, only_total_score=TRUE, print_step=0)

# psi
perf_psi(
  score = list(train = train_score, test = test_score),
  label = list(train = train$y, test = test$y),
  x_limits = c(250, 700), x_tick_break = 50 )
```

谢谢