# **信贷风控模型区分度指标: AUC Gini KS Lift**

在信贷风控领域，我们的目标通常是识别并预测出可能会违约的借款人。为了达到这个目标，我们会构建预测模型，用它来预测每个借款人违约的概率。然而，预测模型的好坏并不是一目了然的，我们需要一些量化的指标来衡量模型的预测性能。这些指标可以帮助我们了解模型的优点和缺点，比较不同模型的性能，以及优化模型的参数。

在这篇文章中，我们将介绍四个在信贷风控中常用的模型性能评估指标：AUC, Gini, KS和Lift。通过学习这些指标，我们可以更全面地评估和理解模型的性能，为信贷风控的决策提供更准确的依据。

## **Toy数据集**

在信贷风控中，我们的目标是区分"好"用户和"坏"用户。通常，我们把没有违约的用户定义为"好"用户，把违约的用户定义为"坏"用户。在模型的训练和评估过程中，我们把"坏"用户视为正样本，把"好"用户视为负样本。

我们准备了一个简单的数据集来进行演示。这个数据集包含了10个用户的评分卡模型预测结果和实际的好坏情况：

|  |  |
| --- | --- |
| 预测结果 | 真实标签 |
| 0.90 | 1 |
| 0.80 | 1 |
| 0.70 | 0 |
| 0.60 | 0 |
| 0.50 | 1 |
| 0.40 | 0 |
| 0.30 | 0 |
| 0.20 | 0 |
| 0.10 | 1 |
| 0.05 | 0 |

在这个数据集中，预测分数表示模型预测每个用户违约的概率，分数越高表示模型预测用户违约的概率越大。实际情况则表示用户是否真的违约，"1"表示用户违约，"0"表示用户没有违约。接下来，我们通过这个小数据集来演示AUC、Gini、KS和Lift的相关计算。

## **AUC**

### **定义**

AUC，全称为Area Under the ROC Curve，指的是ROC曲线下的面积。ROC曲线是以假阳性率（False Positive Rate，FPR）为横轴，真阳性率（True Positive Rate，TPR）为纵轴的曲线。这条曲线反应了好坏用户对中被正确排序的比例。

在信贷风控中，TPR是全体坏用户被模型预测为坏用户的比例，也是大家长在excel里做的随信用分排序的累积坏用户分布统计量，FPR是全体好用户被模型预测为好用户的比例，也是随信用分排序的累积好用户分布统计量。AUC实际上度量的是真实好坏用户按照模型预测结果排序对中，好坏用户被排序正确的比例。

这个比较抽象，我们拿上面构造的数据来举例，我们把好坏用户排成二维矩阵，坏用户排成行，好用户排成列，矩阵里的每个数值是模型的预测结果。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 好\坏 | 0.9 | 0.8 | 0.5 | 0.1 |
| 0.7 | Y | Y | N | N |
| 0.6 | Y | Y | N | N |
| 0.4 | Y | Y | Y | N |
| 0.3 | Y | Y | Y | N |
| 0.2 | Y | Y | Y | N |
| 0.05 | Y | Y | Y | Y |

看第一行第一列，好用户0.7<坏用户0.9，排序正确。第一行最后一列，好用户0.7>坏用户0.1，排序错误。我们总共有6个好用户和4个坏用户，总共有6\*4=24个好坏用户对，其中排序正确的有17对，那AUC就是17/24=0.708。

### **性质**

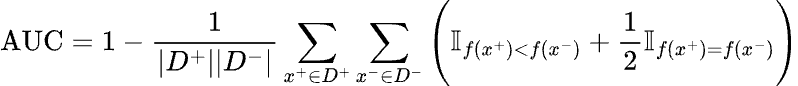
AUC有一些重要的性质：

AUC的取值范围在0.5到1之间。AUC等于0.5表示模型的预测能力不比随机猜测好，AUC等于1表示模型的预测能力完美。

AUC对模型的预测概率的绝对值不敏感，只关心预测概率的排序。这使得AUC能够评估模型的预测性能，而不受阈值的影响。

### **计算**

周志华老师在[西瓜书](https://link.zhihu.com/?target=https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/MLbook2016.htm" \t "/Users/liulebin/Documents\\x/_blank)了给出的AUC计算公式是



这个公式其实就是在计算好坏用户对排序对的比例。我们还是以前面的数据为例，好用户有6个所以 |�+| =6, 坏用户有4个，所以 |�−| =4。 |�+||�−| 就是好坏用户的总对数。 ��(�+)<�(�−) 是模型排错了这个用户对，后面 12��(�+)=�(�−) 是这个好坏用户对序一样记录分错一半。最后用1减去分错的比例就得到了分对的比例。

我们可以根据上面的定义手动编写python代码来计算AUC：

*# 预测分数和实际的好坏情况*scores **=** np**.**array([0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05])actual **=** np**.**array(['坏', '坏', '好', '好', '坏', '好', '好', '好', '坏', '好'])

*# 找出所有的正负样本对*positive\_scores **=** scores[actual **==** '坏']negative\_scores **=** scores[actual **==** '好']pairs **=** [(p, n) **for** p **in** positive\_scores **for** n **in** negative\_scores]

*# 计算模型预测正确的样本对的数量*correct\_pairs **=** sum(1 **for** p, n **in** pairs **if** p **>** n)

*# 计算AUC*auc **=** correct\_pairs **/** len(pairs)print("AUC: ", auc)

也可以直接使用sklearn的包来进行计算：

from sklearn import metrics

# 将实际的好坏情况转换为二进制标签

binary\_actual = (actual == '坏').astype(int)

# 计算AUC

auc = metrics.roc\_auc\_score(binary\_actual, scores)

print("AUC: ", auc)

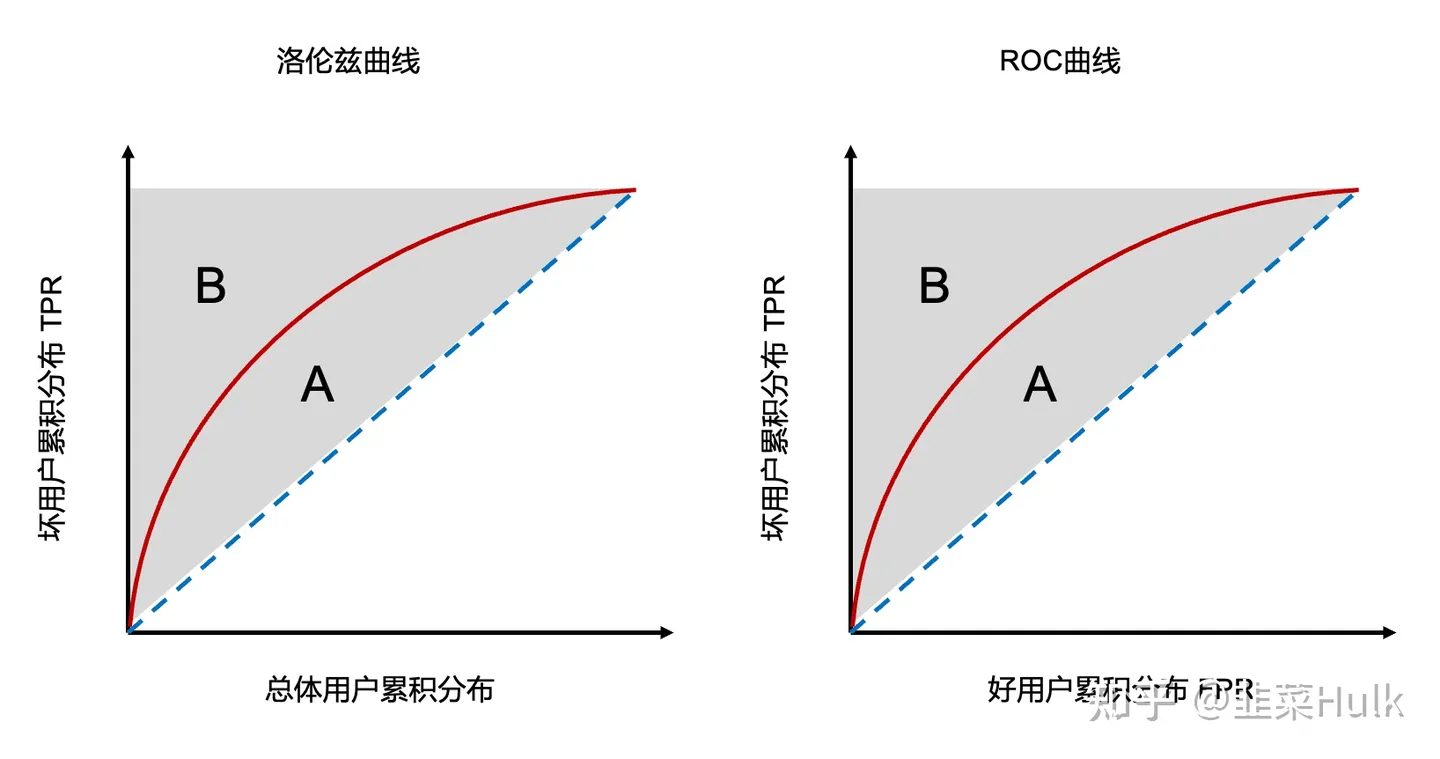
上述两个代码计算出的AUC约为0.708。

## **Gini**

### **定义**

Gini系数衡量的是洛伦兹曲线(Lorenz Curve)或称累积收益曲线（Cumulative Gain Curve）下与对角线之间的两倍面积。如果我们将所有用户按照模型预测违约的概率从高到低排序，然后画出累积曲线（横轴是用户的累积比例，纵轴是坏用户的累积比例即TPR），那么Gini系数就是累积曲线下与对角线之间的两倍面积。

而在风控领域(参考[这本书](https://link.zhihu.com/?target=https://econpapers.repec.org/bookchap/oxpobooks/9780199226405.htm" \t "/Users/liulebin/Documents\\x/_blank))，这个定义被改动了，把在洛伦兹曲线下做计算换成了ROC曲线下做计算。其实这两个曲线的纵坐标都是坏用户累积分布TPR，只是洛伦兹曲线的横坐标是总体累积分布，而ROC的横坐标是好用户的累积分布。这个改动其实不影响评估的准确性。只是参考标准换了下而已。



### **性质**

Gini系数的取值范围在-1到1之间，0表示模型预测无好于随机猜测，1则表示模型预测完全正确。它是一种常用的模型性能评估指标，可以帮助我们理解模型在区分正负样本方面的能力。

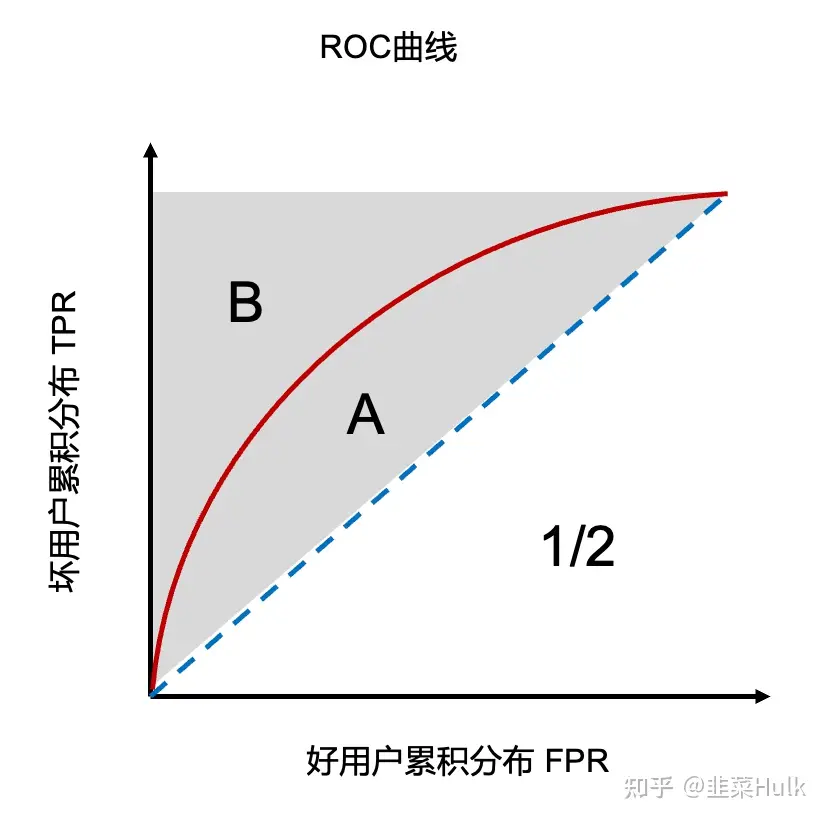
需要注意的是，Gini系数与机器学习中经常使用的基尼不纯度（Gini Impurity）是两个不同的概念。基尼不纯度是用来衡量数据的混乱程度的，其值越大，数据的混乱程度越高，通常在决策树和随机森林等树模型中做叶子生长判断时使用。而Gini系数则是用来衡量模型区分度的。

### **计算**

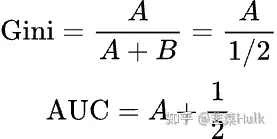
在计算Gini系数的时候通常会用到它和AUC之间的等式关系：

IMG_259

这个关系可以简单的用图形关系推导出来。



根据定义，Gini为图中A部分的面积除以A+B的面积。而AUC是ROC曲线下的面积即A部分的面积再加上下面三角形区域即1/2(横纵坐标最大值为1，正方形总面积为1，下三角形面积是正方形面积的一般)。



可以导出 Gini=AUC−1/21/2=2AUC−1 。

我们通过python代码，先按照面积定义手算一遍Gini系数的值，然后再用2AUC-1的公式验证数值的正确性。实际工作中，直接用2AUC-1来计算即可。

import numpy as np

from sklearn.metrics import roc\_auc\_score

# Predicted credit risk scores and actual binary labels

predict = [0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05]

actual = [1, 1, 0, 0, 1, 0, 0, 0, 1, 0]

def compute\_gini\_coefficient(predict, actual):

# Combine the predicted scores and actual labels into a single array

data = np.array(list(zip(predict, actual)), dtype=[('predict', float), ('actual', int)])

# Sort the data based on the predicted scores (in descending order)

data = np.sort(data, order='predict')[::-1]

n = len(data)

# Calculate the cumulative share of positive (good) labels

cumulative\_pos = np.cumsum(data['actual']) / np.sum(data['actual'])

cumulative\_neg = np.cumsum(1-data['actual']) / np.sum(1-data['actual'])

# Calculate the area between Lorenz curve and perfect equality

area\_between\_curve = np.trapz(cumulative\_pos, cumulative\_neg) - 0.5

# Calculate the area under perfect equality

area\_under\_curve = 0.5

# Compute the Gini coefficient

gini\_coefficient = area\_between\_curve / area\_under\_curve

return gini\_coefficient

# Compute Gini coefficient

gini\_coef = compute\_gini\_coefficient(predict, actual)

# Compute AUC

auc = roc\_auc\_score(actual, predict)

# Verify Gini = 2 \* AUC - 1

gini\_verified = 2 \* auc - 1

print(f'Gini Coefficient: {gini\_coef:.4f}')

print(f'AUC: {auc:.4f}')

print(f'Gini (Verified): {gini\_verified:.4f}')

plt.plot(cumulative\_neg, cumulative\_pos,color='r')

plt.plot([0, 1], [0, 1], linestyle='--', color='b')

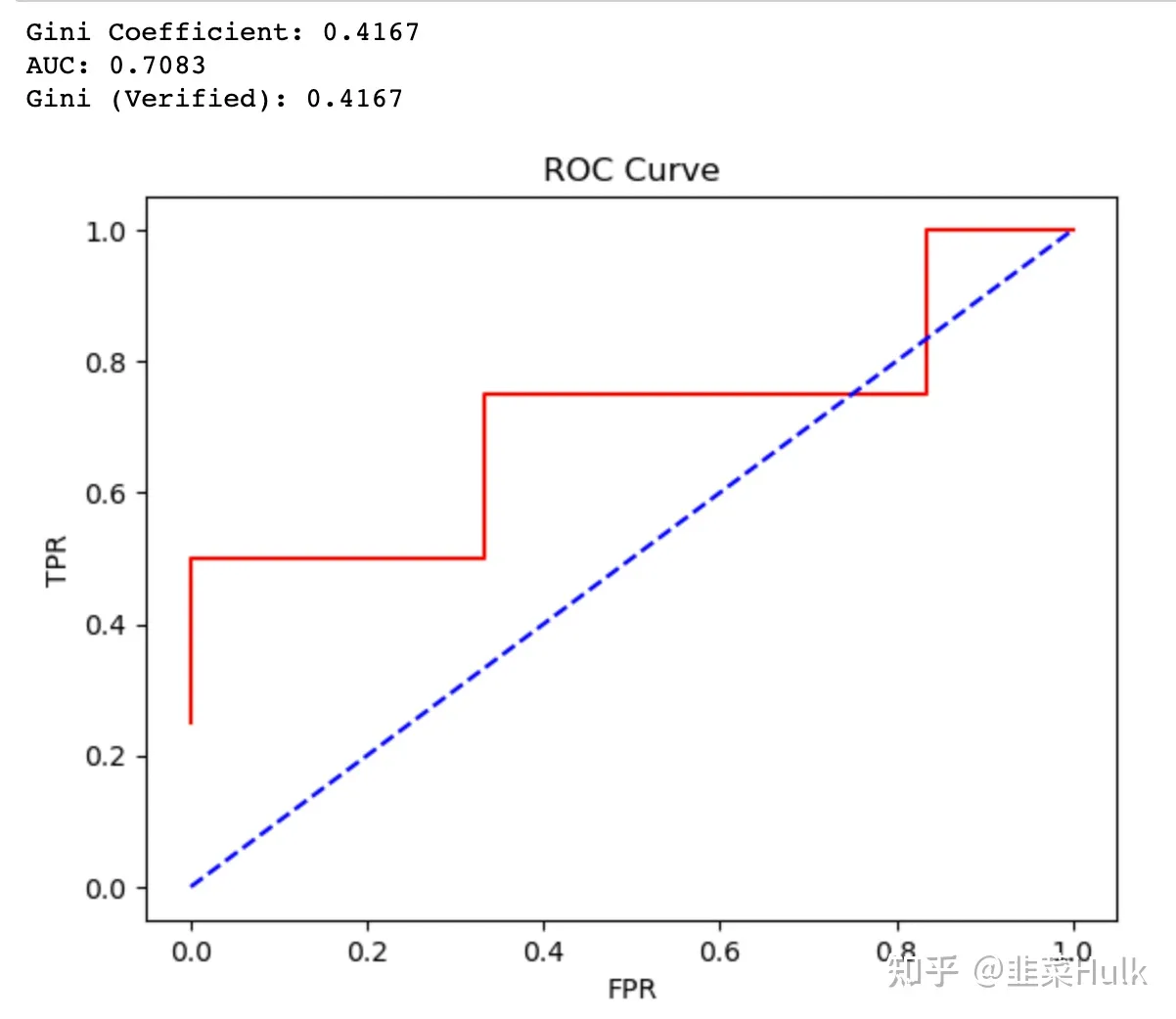
plt.xlabel('FPR')

plt.ylabel('TPR')

plt.title('ROC Curve')

plt.show()

得到的结果如下：



## **KS**

### **定义**

在信贷风控中，KS(Kolmogorov-Smirnov)指标用于衡量模型区分能力的强弱。具体来说，KS值是好坏用户累积分布曲线之间的最大差值，即 max(abs(TPR-FPR))，表示模型在对好坏用户进行分类时的最大预测准确度。

### **性质**

KS值的取值范围是0到1，数值越大表示模型的区分能力越强。当KS值等于1时，模型完美地区分了好用户和坏用户，即好用户和坏用户的累积分布曲线没有重叠。反之，当KS值接近于0时，模型无法有效地区分好用户和坏用户，即好用户和坏用户的累积分布曲线几乎重合。

### **计算**

我们还是用手动方式和调用sklearn包的方式来计算KS的值。

手动计算的步骤如下：

1. 将数据根据模型预测分数从0到1排序。
2. 计算好坏用户累积数量。
3. 计算好用户和坏用户的累积分布，即FPR和TPR。
4. 计算好用户和坏用户的累积分布之间的最大差值，即KS值。

代码如下：

import numpy as np

# 数据和标签

scores = np.array([0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05])

actual = np.array(['坏', '坏', '好', '好', '坏', '好', '好', '好', '坏', '好'])

binary\_actual = (actual == '坏').astype(int)

# 手动计算KS值

n = len(binary\_actual)

sorted\_indices = np.argsort(scores)[::-1]

sorted\_actual = binary\_actual[sorted\_indices]

cum\_TP = np.cumsum(sorted\_actual == 1) / np.sum(sorted\_actual == 1)

cum\_FP = np.cumsum(sorted\_actual == 0) / np.sum(sorted\_actual == 0)

# 计算好用户和坏用户的累积分布曲线

cum\_bad = cum\_TP

cum\_good = cum\_FP

# 计算KS值

ks\_manual = np.max(np.abs(cum\_good - cum\_bad))

print("KS (手动计算): ", ks\_manual)

按照 KS=max(|FPR−TPR|) 的公式调用sklearn的包进行计算：

from sklearn.metrics import roc\_curve

# 使用sklearn包计算KS值

fpr, tpr, \_ = roc\_curve(binary\_actual, scores)

ks\_sklearn = np.max(np.abs(tpr - fpr))

print("KS (sklearn包计算): ", ks\_sklearn)

## **Lift**

### **定义**

提升度指标Lift是用于衡量模型在目标用户中的区分能力。Lift值是累积响应曲线（Cumulative Response Curve）和随机响应曲线（Random Response Curve）之间的差异。它表示模型对于目标用户的预测能力相对于随机预测的提升程度。这个指标其实更常出现在营销响应模型的评估中，但风控领域也是可以作为一个参考指标使用。

在风控领域，Lift其实就是坏用户累积分布和总体用户的累积分布的比值。他刻画的信息和KS是有相似之处的。

### **性质**

Lift值大于1：表示模型在目标用户中的预测能力优于随机预测。例如，Lift值为2表示模型的预测效果是随机预测的2倍，即模型在目标用户中的预测准确率较高。

Lift值等于1：表示模型在目标用户中的预测能力与随机预测相当。例如，Lift值为1表示模型的预测效果与随机预测相同，即模型没有提供额外的预测能力。

Lift值小于1：表示模型在目标用户中的预测能力不如随机预测。例如，Lift值为0.5表示模型的预测效果是随机预测的一半，即模型在目标用户中的预测准确率较低。

### **计算**

依旧是用手动方式和调用sklearn包的方式来计算KS的值。

手动计算Lift值的步骤如下：

1. 将数据按模型预测结果从0到1排序。
2. 分别计坏用户的累积数量和总用户的累积数量。
3. 分别计坏用户的累积分布和总用户的累积分布。
4. 坏用户累积分布除以总用户累积分布得到Lift。

代码如下:

import numpy as np

# 数据和标签

scores = np.array([0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05])

actual = np.array(['坏', '坏', '好', '好', '坏', '好', '好', '好', '坏', '好'])

binary\_actual = (actual == '坏').astype(int)

# 手动计算Lift

n = len(binary\_actual)

sorted\_indices = np.argsort(scores)[::-1]

sorted\_actual = binary\_actual[sorted\_indices]

cum\_TP = np.cumsum(sorted\_actual == 1) / np.sum(sorted\_actual == 1)

cum\_Total = np.cumsum([1]\*len(sorted\_actual)) / len(sorted\_actual)

lift = cum\_TP/cum\_Total

print(lift)

使用sklearn的相关包计算，这里我们使用的不是sklearn的包而是scikit-plot。

import numpy as np

# 数据和标签

scores = np.array([0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05])

actual = np.array(['坏', '坏', '好', '好', '坏', '好', '好', '好', '坏', '好'])

binary\_actual = (actual == '坏').astype(int)

# 使用scikit-plot中计算累积收益曲线的函数计算

cumulative\_gains, gains = skplt.metrics.cumulative\_gain\_curve(binary\_actual, scores)

deciles = np.linspace(0, 1, num=len(cumulative\_gains))

# 剔除数组最开始的0 然后计算lift

lift = tpr[1:]/deciles[1:]

print(lift)

## **小结**

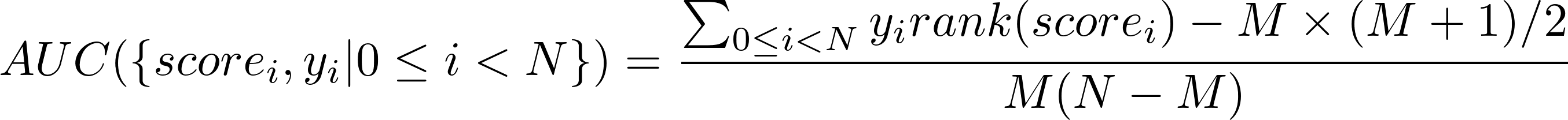
在信贷风控领域，评估和优化模型的性能是非常重要的，这涉及到多个指标的使用。本文介绍了在信贷风控中常用的四个重要指标：AUC、Gini、KS和Lift。其中AUC和Gini都是反应了模型区分度的平均状况，他们可以认为是同一个指标。KS和Lift都是在看累积分布的差异，KS更关注累积分布差异的最大值，而Lift反应差异的变化趋势。

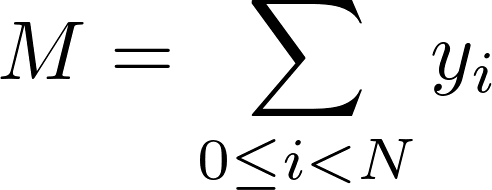
**回归建模序准评估**

核心指标

1. 订单数

订单数可认为是0/1二分类目标，采用AUC作为序准的离线评估指标



，即M为正样本个数

2. GMV

gmv是回归目标，采用gini score作为离线评估指标, giniscore是洛伦兹曲线的AUC，是二分类auc目标在回归问题上的扩展，可用于评估回归问题的序准

