第四章 银行业数据分析

方杰、李烜

福建江夏学院金融学院

本章内容

1. 银行经营绩效的线性回归分析
   - 回归分析的概念和原理
   - 线性回归模型的应用
2. 银行信贷风险的逻辑回归分析
   - 信贷风险的逻辑回归模型

回归分析的概念

回归分析: 确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法

一元线性回归的数学模型

$$y = ax + b + \varepsilon$$

多元线性回归的数学模型

$$y = a_1x_1 + a_2x_2 + \cdots + a_nx_n + b + \varepsilon$$

- y: 因变量
- b: 常数项
- ε: 误差项
- x_i: 自变量 $(i = 1, 2, \ldots, n)$
- a_i: 自变量系数 $(i = 1, 2, \ldots, n)$

线性回归模型的原理

残差平方和 SSE，用作衡量实际值与模型估计值的接近程度

$$\mathrm{SSE}=\sum_i\left(y^{(i)}-\hat{y}^{(i)}\right)^2=\sum_i\left[y^{(i)}-\left(ax^{(i)}+b\right)\right]^2$$

最小二乘法（Least Squares）的目标是使得残差平方和最小，此时的 a 和 b 即为所求的线性回归模型回归系数和常数项。

方法：对残差平方和求导，令导数为 0，求解 a 和 b

线性回归模型的应用

线性回归的基本步骤：

- 确定自变量和因变量
- 变量间相关性分析
- 建立回归分析模型
- 模型检验

确定自变量和因变量

- 明确考察的具体目标——因变量
- 目标的相关影响因素——自变量

变量间相关性分析

回归分析：对具有因果关系的影响因素 (自变量) 和考察对象 (因变量) 所进行的数理统计分析处理。当变量与因变量确实存在某种关系时，建立的回归方程才有意义。

相关系数：变量间具有较高相关性，说明建立线性模型进行分析是具有实际意义的。

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}\left[(X-\mu_X)(Y-\mu_Y)\right]}{\sigma_X \sigma_Y}$$

Python 代码实现

```
df.corr()    # 生成的结果是相关性矩阵
```

建立回归分析模型

statsmodels库中的ols函数可以建立回归分析模型

```
import statsmodels.formula.api as sm
import pandas as pd
result = sm.ols('Y ~ X', dataset).fit()
```

- Y: 指定线性回归的因变量
- X: 指定线性回归的自变量
- dataset: Y 和 X 所在的 DataFrame 格式的数据
- fit(): 用于对模型进行拟合
- result: 模型结果，包括自变量的系数和常数项等。可以使用result.params调用模型中的结果

回归分析的结果

使用以下命令，可得到回归分析结果result的表格

```
print(result.summary())
```

- Dep. Variable: 输出变量，因变量
- Model: 使用的模型类型
- Method: 模型求解方式
- No. Observations: 观测值的数量
- Df Residuals: 残差计算的自由度
- coef: 回归系数，其中Intercept为常数项（截距项）
- std err: 常数项和变量系数的估计标准差
- t, P>|t|: 分别为常数项和变量系数的 t 统计量值，以及 t 检验下的 p-value
- [0.025, 0.075]: 95% 显著性水平下的置信区间

回归分析的结果 (cont.)

- Skew, Kurtosis: 分别为残差的偏度和峰度，可以用来判断残差是否符合正态分布
- Durbin-Watson: 判断残差是否一阶相关
- R-squared: 拟合优度，取值越接近于 1，表示拟合效果越好
- Adj. R-squared: 通过样本数量和模型参数数量修正后的拟合优度
- F-statistic, Prob(F-statistic): F 统计量及原假设成立概率
- Log-Likelihood: 对数极大似然估计
- AIC, BIC: 赤池信息准则和贝叶斯信息准则，用于判断最优模型
- Jarque-Bera(JB), Prob(JB): Jarque-Bera 统计量及原假设成立概率，用于残差正态性的判断

模型检验

为观察拟合效果，输入绘制代码：

```
import matplotlib.pyplot as plt
# 绘制散点图
plt.scatter(data['X'], data['Y'])
# 拟合线性回归图
plt.plot(data['X'], result.params[0]
         + result.params[1]*data['X'], 'r')
plt.text(1,3, 'y=' + str(round(result.params[1],4)) + '+'
         + str(round(result.params[0],4)) +'*x')
plt.title('linear regression')
plt.show()
```

信贷风险评估

商业银行信贷风险评估，主要是在客户贷款之前，然而在发放信贷过程中或者之后的风险，银行很难及时发现和控制，需要客户在与银行发生业务过程记录的信息来评价贷后风险，以利于及时作出风险管控措施。

信贷违约风险

违约概率 (Probability of default) 是预计债务人不能偿还到期债务 (违约) 的可能性。在实践中，银行会将违约概率对作为量化信用风险的关键参数之一，并依此对用户分类。

在实践中，银行会将次级、可疑、损失三种类型列为信贷违约。

传统违约概率计算模型

- CreditMetirics 模型
- KMV 模型
- Credit Risk 模型

传统风险度量方式弊端

- 模式只适用于经营管理、信息记录规范的企业
- 模型不全面，忽略了一些市场风险，与市场真实情况存在一定误差
- 信息不对称，使用的公司数据非实时，结果可靠性有限

信贷风险的逻辑回归模型

受限因变量模型（Limited Dependent Variable model, LDV），是一类因变量取值为二分类数据的特殊模型。常见的有 Logit 模型、Probit 模型等。

逻辑（Logistic）回归（Logit Regression）是目前银行建立大数据信用评估最常用的方法，可以解决二分类的非线性问题。

在信用评估时，用逻辑回归模型返回的违约概率结果是 0 到 1 之间以小数形式呈现的类概率数字，用以判断客户是否会违约，以及违约的可能性大小，具有对特征可解释性强的优点。

逻辑回归的数学模型

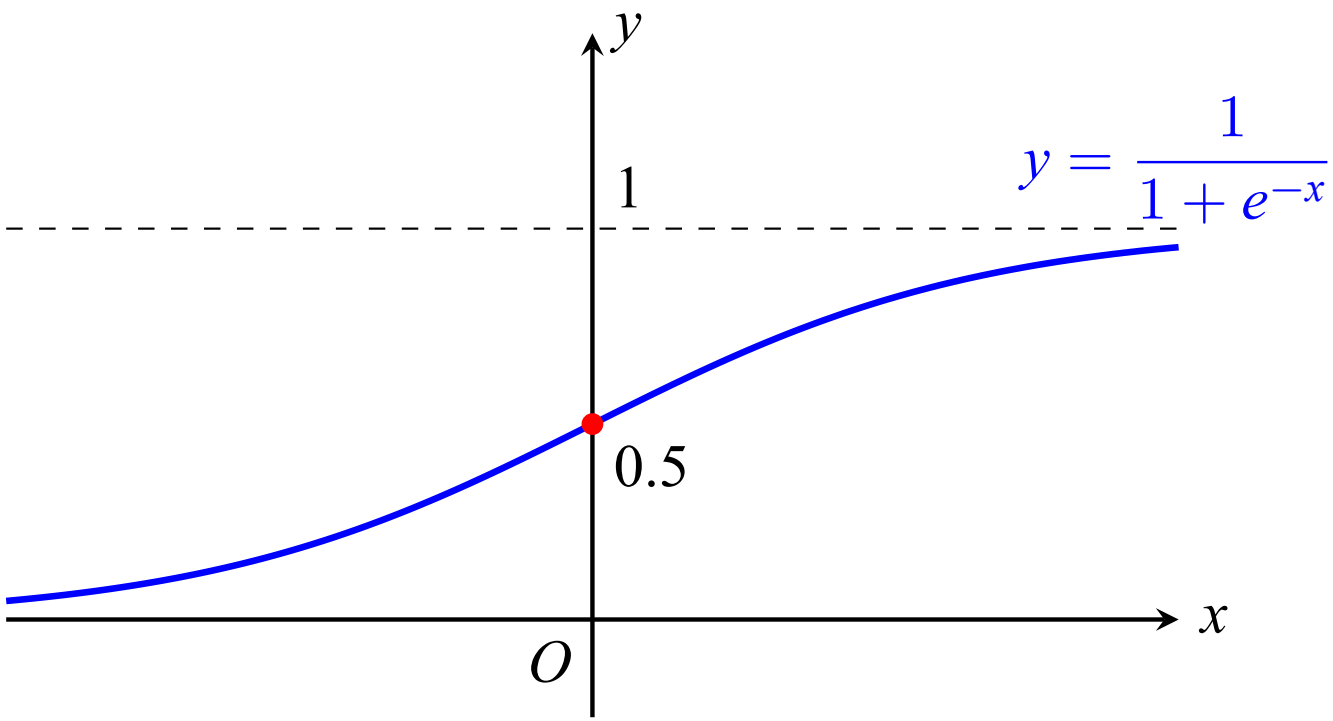
假设线性回归模型

$$\mathbf{a}'\mathbf{x}+b=\begin{bmatrix}a_1 & a_2 & \cdots & a_n\end{bmatrix}\begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix}+b=a_1x_1+a_2x_2+\cdots+a_nx_n+b$$

然后使用一个单位阶跃函数，将结果映射到 $[0,1]$。对于逻辑回归模型而言，该函数的形式如下：

$$f(\mathbf{x})=\frac{1}{1+e^{-(\mathbf{a}'\mathbf{x}+b)}}$$

单位阶跃函数



逻辑回归的数学模型 (cont.)

将逻辑回归输出与二分类问题输出 $y = \{0, 1\}$ 建立联系

设 0.5 为临界值:

- 当 $f(\mathbf{x}) > 0.5$，即 $\mathbf{a}'\mathbf{x} + b > 0$ 时，y 为 1;
- 当 $f(\mathbf{x}) < 0.5$，即 $\mathbf{a}'\mathbf{x} + b < 0$ 时，y 为 0;
- $f(\mathbf{x})$ 越接近临界值 0.5 则无法判断，分类准确性下降。

说明:

逻辑回归被广泛应用于分类问题，并且不要求自变量和因变量之间有线性关系。与线性回归模型相比，逻辑回归对异常值不敏感

逻辑回归分析的 Python 实现

Python 中statsmodels包的Logit函数可以实现逻辑回归模型构建

```
import numpy as np
from statsmodels.api import Logit
logit = Logit(yTrain, XTrain)
# yTrain是因变量(取值为0或1)训练集; XTrain是自变量训练集
result = logit.fit()
print(result.summary())
```

逻辑回归分析的预测与检验

用测试集进行预测，结果为事件发生的概率，取值在 $[0,1]$ 之间。将预测结果与 0.5 进行比较，大于等于 0.5，则判定事件发生，取值为“1”；反之，判定事件不发生，取值为“0”

```
import pandas as pd
pred = result.predict(XTest)
# result是前面逻辑回归模型的拟合结果
Value = round(pred) # 预测结果>=0.5的，判定为1；反之为0
compare = pd.DataFrame({'predictedValue':pred, 'roundValue'
    :Value, 'actual':y.tail(10)}) # 引号中的是列名定义
print(compare) # 这里y.tail(10)存储的是实际因变量取值
print(np.sum(Value==y.tail(10))/10) # 计算10个预测值的准确率
```