内容简介

在完成数据准备后，本节通过逻辑回归模型，对公司是否会发生交易进行分类，回答本案例中的业务问题。

1. 主要内容

- 逻辑回归模型、混淆矩阵和 ROC 曲线的概念；
- 逻辑回归模型的构建、训练、测试、评估；
- 网格搜索超参数；
- 使用混淆矩阵和 ROC 曲线评估模型的分类效果；
- 输出模型结果并对结果进行分析

2. 学习目标

学完本节，能解决以下问题：

- 如何构建逻辑回归模型并完成从训练模型、测试模型到评估模型的过程？
- 如何使用网格搜索法搜索本案例逻辑回归模型的最优的超参数组合？
- 如何使用混淆矩阵和 ROC 曲线评估模型的分类效果？
- 如何输出本案例的分析结果并对结果进行解读?

逻辑回归模型

逻辑回归属于对数线性回归，分为二分类和多分类的回归模型。一般二分类逻辑回归模型应用场景较多，下文逻辑回归模型均指二分类逻辑回归模型。我们期望使用广义线性模型实现分类任务，对输入的 X 建立线性回归式，再将分类任务的标记 Y 和输入 X 的线性模型联系起来，最终输出 Y 属于某一类别的概率。逻辑回归要求连续变量服从类似正态分布。

逻辑回归模型中的因变量（y）的只有 1/0（如：是/否、发生/不发生）两种取值，假设在 p 个独立自变量 $x_1, x_2, \cdots, x_p$ 作用下：

记 y 取值为 1 时的概率： $P(y=1|X)$

记 y 取值为 0 时的概率： $P(y=0|X)$

因为 y 取值只能是 1 或 0，因此 $P(y=1|X)+P(y=0|X)=1$

对取 1 和取 0 的概率之比求自然对数 $\ln\left(\frac{p(y=1|x)}{p(y=0|x)}\right)$

令其等于线性回归函数 $z=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_px_p$

即：

$$\ln\left(\frac{p(y=1|x)}{p(y=0|x)}\right)=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_px_p$$

逻辑回归模型的功能就是给定已知的 x 和 y，通过不断调整权重的取值，优化模型，找到最理想情况下不同特征对应的权重 β 以及 β_0，从而使得模型的分类效果最佳。

通过转换可以求得取 1 和取 0 时分别的概率：

$$P(y=1|X)=\frac{1}{1+e^{-z}}$$（该函数称为 sigmoid 方程）

$$P(y=0 \mid X) = \frac{1}{1+e^{z}}$$

逻辑回归的结果只是一个概率，如何根据概率来判断某一样本属于哪一类则需要确定阈值。高于某一概率阈值的判定为 1，否则判定为 0。

模型优化的方法主要是通过迭代调整权重，从而使得损失函数的值最小。

损失函数是一类用于评估模型的预测值 f(x)与真实值 Y 的不一致程度的函数，损失函数的值越大，表明预测值与实际值的差别越大，即现一轮迭代中的权重不是最理想的情况。

最常见的一种损失函数为最小二乘法，函数的公式为

$$\sum_{i=1}^{n}(Y - f(X))^2$$

Y ：真实值　　$f(X)$ ：回归线的预测值

逻辑回归具有可稀疏学习的特性。

稀疏指的就是有较多特征的权重为 0。

具体而言，就是通过在损失函数中引入正则化项让不够显著的特征对应的权重趋近于零，保留下意义更重要的非零参数对应的特征。

正则化

当模型的参数过多时，会导致模型的复杂度大大提高，从而模型过度解读现有数据集，而丧失一般性。正则化是通过“惩罚”部分参数，来控制模型的复杂度的一种方法。

本案例调用的是 sklearn 中的 LogisticRegression 函数，该函数共有 14 个参数。

混淆矩阵和 ROC 曲线

对于分类算法而言，较常使用的评估方法有构建混淆矩阵，绘制 ROC 曲线。

1. 混淆矩阵

混淆矩阵也称误差矩阵，主要用于比较分类结果与实际测得值间的差异，可以把分类结果的精度显示在一个混淆矩阵里面。

二分类问题结果的混淆矩阵如下表所示：

	预测为 0	预测为 1
真实为 0	TN	FP
真实为 1	FN	TP

真阴(True Negative)：真实为 0，预测也为 0；

假阳(False Positive)：真实为 0，预测为 1；

假阴(False Negative)：真实为 1，预测为 0；

真阳(True Positive)：真实为 1，预测也为 1。

根据以上四种样本的分类，可以得到对于分类模型的不同评价指标。最常见的如：

准确度(accuracy)=(TP+TN)/(TP+FP+FN+TN)；

精确度(precision)=TP/(TP+FP)；

召回率(recall)=TP/(TP+FN)。

本案例中通过**准确率**来评估模型在测试集上的表现。

注：测试集是非平衡数据集，sklearn 包中 balanced_accuracy_score()函数针对非平衡数据集。

目前关于 accuracy 和 precision 的中文名称还不是很统一，一些地方会将 accuracy 称为精确度，而把 precision 称为准确度，我们秉承以记忆英文名称与对应计算公式为主的原则，暂统一二者为：accuracy-准确度，precision-精确度。

2. ROC 曲线

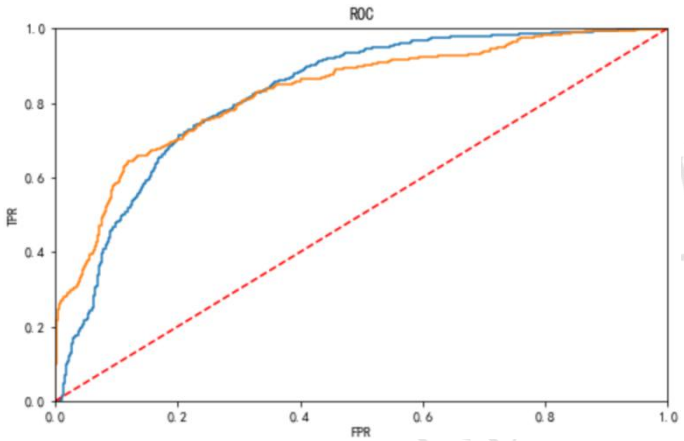
又称为接收者操作特征曲线（receiver operating characteristic curve），说明了

在不同阈值设定下该模型对两类的判断正误程度。

ROC 曲线的绘制由每个阈值下的假阳率和真阳率组成，其中：

假阳率 FPR = FP / (FP + TN)；

真阳率 TPR = TP / (TP + FN)。



我们可以看出：左上角的点（FPR=0、TPR=1），为完美分类 ROC 曲线。因此，曲线距离左上角越近，证明分类器效果越好。

结果分析

通过网格搜索超参数后得到本案例逻辑回归模型最优的参数组合，同时输出零特征占比和测试准确率，以及非零特征以及这些特征对应的系数，仅有 11 个特征被保留下来，用于预测是否发生交易。如下图所示：

```
零特征占比 Sparsity: 72.50%
测试准确率 Test score: 72.63%
以下为非零特征及对应系数：
          feature_name              coef
 transfer_mode_x0_做市            0.4075
         listing_days             0.5312
  registered_capital              0.2649
               asset              0.0911
           maker_num              0.8824
 net_asset_per_share              0.0128
   ave_executive_age             -0.1232
       holder_rt_1st             -0.3807
            asset_gr              0.0225
          income_gr               0.0195
  contain_org_holder              0.0242
```

分析结果输出图

经过整理这 11 个字段分类为下表所示：

非零特征分析表

类型	英文名称	中文名称	权重系数
负向作用字段	holder_rt_1st	第一大股东占比	-0.3807
	ave_executive_age	高管平均年龄	-0.1232
明显正向作用字段	Registered_capital	注册资金	0.2649
	Transfer_mode_x0_做市	做市转让方式	0.4075
	Listing_days	挂牌时长	0.5312
	Maker_num	做市商数量	0.8824
一般正向作用字段	net_asset_per_share	每股净资产	0.0128
	income_gr	营业收入增长率	0.0195
	asset_gr	总资产增长率	0.0225
	contain_org_holder	存在机构股东	0.0242
	Asset	资金总计	0.0911

注：标黄为新发现特征；
具体参数的数值并不具有完全的可比性。
前文进行数据变换将大部分数据变成标准正态分布方便参数具有可比性，但是由于原始分布并不完全相同，参数之间仍然存在微小的差异。解读时参考绝对值大小和正负向，但不可以比较。

（1）负向作用字段

holder_rt_1st 表明第一大股东占比越高，交易可能性越低。可能存在有与上述类似的情况。第一大股东占比较高，则他对公司的控制权相对较高。一方面其本人可能会把持股票，另一方面许多投资人可能不愿意通过交易的方式介入公司成为小股东受到风险。

ave_executive_age 较高时发生交易几率下降。这是一个全新的发现，即股东高管年龄较高的时候，交易更难以发生。可能存在高管股东较为保守、把持股票控制权、避免交易产生的风险和稀释作用。

（2）明显正向作用字段

registered_capital 具有明显的正向作用，反应的是公司本身的素质。注册资本越多，交易越有可能发生。这个指标与体量有一定关联，注册资本表示公司起步的水平较高。

transfer_mode_x0_做市指的是转让方式 transfer_mode 字段中取值为“做市”的情况，另一种转让方式为“协议”。做市制度要求公司必须交易，而协议制度中公司可以不发生交易。这也说明了模型对现实具有一定吻合性。

listing_days 挂牌时间对交易有正向作用。有两种可能的原因。第一种，假设每天交易的可能性相同，时间越长，所有天交易的可能性本身就会上升；第二种，挂牌时间较短的公司，尚未准备好接受市场交易，仍有待推进制度建设。现实种可能两种原因均存在，而后者的可能性更大。

maker_num 做市商数量对于交易有正向作用。做市商在新三板市场流动性中起到

了积极作用。

（3）一般正向作用字段

其他的特征 net_asset_per_share、income_gr、asset_gr、contain_org_holder、asset 表明了公司具有正向增长或者规模更大时，发生交易可能性更高。其中 contain_org_holder 反映了公司股东的一种特殊性质，其与是否发生交易的正向关系是一种比较新的发现。

逻辑回归总结

1、如何构建逻辑回归模型并完成从训练模型、测试模型到评估模型的过程？

（1）构建逻辑回归模型

首先从 sklearn 中导入逻辑回归模型，代码：

from sklearn.linear_model import LogisticRegression

其次创建逻辑回归模型并设置其中 5 个参数，例如案例中的代码：

clf = LogisticRegression(penalty='l1',solver='saga',tol=0.1,C=0.1,random_state=0)

penalty 是正则化选择参数，当 penalty=l1 时，表示倾向于稀疏化的模型参数结果，当 penalty=l2 时，没有这种倾向；

solver 是优化算法选择参数，这个参数决定了我们对逻辑回归损失函数的优化方法，不同的损失函数适合使用不同的优化方法，saga 方法的适用情况更普遍；

tol 是停止求解的标准，代表收敛阈值。数值越小，要求收敛的稳定性越高，需要的收敛时间越长；

C 是用于调整正则化的力度，较小的值表示正则化力度更强，结果将更加稀疏。

以及随机状态参数 random_state。

设置了这 5 个参数后，将创建的逻辑回归模型命名为 clt。

（2）训练模型

使用训练集开始训练模型，训练模型的函数为 fit()，例如案例中的代码：

clf.fit(X_train_bal, y_train_bal)

X_train_bal, y_train_bal 是本案例中经过重采样得到的平衡数据集。

（3）测试模型

使用测试集测试训练好的模型，使用 predict()函数得到该模型的预测分类结果，例如案例中的代码：

y_pred = clf.predict(X_test)

X_test 为测试集中样本的特征，将得到的结果命名为 y_pred

（4）评估模型

评估模型的指标有两个：平衡准确度和稀疏度

①平衡准确度是从数值上直观反正模型效果的指标；

②稀疏度是反映逻辑回归模型中权重系数为零的特征占所有分类特征的比例，值越高，参与分类的特征越少，模型的复杂度就越低，反之则复杂度越高。

这里测试集的数据也是非平衡数据，对于非平衡数据集需要采用平衡准确度 balanced_accuracy_score()函数来计算模型的平衡准确度，例如案例中的代码：

```
# 导入平衡准确度函数
from sklearn.metrics import balanced_accuracy_score
# 评估它在测试集上的表现：计算平衡准确度
balanced_accuracy_score(y_test, y_pred)
```

```
0.739044433655212
```

经过计算，本案例训练的模型平衡准确率约等于 0.74。

对于稀疏度，计算方法是：

```
# 评估它在测试集上的表现：计算稀疏度
np.mean(clf.coef_ == 0)
```

```
0.025000000000000001
```

得到该模型的稀疏度约等于 0.025，即有 97.5%的分类特征被保留下来，模型复杂度非常高。

2、如何使用网格搜索法搜索本案例逻辑回归模型的最优的超参数组合？

超参数是指模型在开始学习过程之前已经设置好取值的参数，而不是通过训练得到的参数数据，通过不断调整超参数，可以实现模型的不断优化。

人为修改其中某个参数或许可以在不断的工作中实现效果，但是如果多个参数之间同时变化，工作量就会以幂次方的倍增。网格搜索超参数就是为了解决这个工作效率而产生的方法。

网格搜索超参数主要包含三步操作：

①新建 DataFrame 保存搜索结果

例如案例中代码：

result_frame = pd.DataFrame(columns=['C', 'tol', 'score', 'sparsity'])

创建 DataFrame ,将其命名为 result_frame ,因为本案例中参数 penalty 和 solver 已经事先设置好 ,需要搜索的超参数分别是 C 和 tol ,同时记录对应的平衡准确度 score 和稀疏度 sparsity。

当得到不同参数 C、tol 下模型的 score、sparsity，只需要其值保存在 DataFrame 中即可。例如案例中的代码：

result_frame=result_frame.append({'tol':tol,'C':C,'score':score,'sparsity':sparsity}, ignore_index=True)

②穷举 C 和 tol 的组合

使用 for 参数 in []的形式分别穷举不同的参数 C 和 tol 取值。例如案例中的代码：

```
# 穷举C和tol的组合
for C in [0.1, 0.01, 0.005, 0.001, 0.0005, 0.0001]:
    for tol in [0.1, 0.01, 0.001, 0.0001]:
```

③使用不同的参数组合，重新训练模型并得到对应的平衡准确度 score 和稀疏度

sparsity。

案例中网格搜索超参数的完成代码如图所示：

```
# 新建DataFrame保存搜索结果
result_frame = pd.DataFrame(columns=['C', 'tol', 'score', 'sparsity'])

# 穷举C和tol的组合
for C in [0.1, 0.01, 0.005, 0.001, 0.0005, 0.0001]:
    for tol in [0.1, 0.01, 0.001, 0.0001]:
        # 创建逻辑回归模型
        clf = LogisticRegression(penalty='l1', solver='saga',
                                 tol=tol, C=C, random_state=0)
        # 训练模型
        clf.fit(X_train_bal, y_train_bal)
        # 跳过没有收敛的模型
        if clf.n_iter_ >= clf.max_iter:
            continue
        # 测试模型
        y_pred = clf.predict(X_test)
        # 评估模型
        score = balanced_accuracy_score(y_test, y_pred)
        sparsity = np.mean(clf.coef_ == 0)
        # 将参数搜索结果放入结果表中
        result_frame = result_frame.append({'tol':tol, 'C':C, 'score':score,
                                            'sparsity':sparsity}, ignore_index=True)
```

其中需要对模型的收敛情况进行设置

为什么会存在收敛问题呢？

逻辑回归模型是一个迭代的过程，在损失函数的约束，模型不断的去迭代循环，以寻求分类特征系数取值的最优效果，而当迭代次数大于 100 时便认为无法收敛，此时计算的指标并不具参考性，因此调过没有收敛的模型超参数组合。

在逻辑回归模型中，迭代次数：n_iter，最大迭代次数：max_iter（默认为 100）。

最后可以使用 print()将结果打印出来，也可以直接在单元格中直接输入 result_frame 查看结果。

其中无法收敛模型警告如图所示：

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\sag.py:334: Convergence
Warning: The max_iter was reached which means the coef_ did not converge
  "the coef_ did not converge", ConvergenceWarning)
```

案例中不同参数设定下不同指标的运行结果如图所示：

以下为不同参数设定下不同指标的运行结果：

	C	tol	score	sparsity
0	0.1000	0.1000	0.734544	0.025
1	0.1000	0.0100	0.744268	0.150
2	0.0100	0.1000	0.728543	0.500
3	0.0100	0.0100	0.727774	0.650
4	0.0100	0.0010	0.727028	0.700
5	0.0050	0.1000	0.718046	0.600
6	0.0050	0.0100	0.712791	0.800
7	0.0050	0.0010	0.714288	0.850
8	0.0050	0.0001	0.714288	0.875
9	0.0010	0.1000	0.500000	1.000
10	0.0010	0.0100	0.500000	1.000
11	0.0010	0.0010	0.500000	1.000
12	0.0010	0.0001	0.500000	1.000
13	0.0005	0.1000	0.500000	1.000
14	0.0005	0.0100	0.500000	1.000
15	0.0005	0.0010	0.500000	1.000
16	0.0005	0.0001	0.500000	1.000
17	0.0001	0.1000	0.500000	1.000
18	0.0001	0.0100	0.500000	1.000
19	0.0001	0.0010	0.500000	1.000
20	0.0001	0.0001	0.500000	1.000

3、如何使用混淆矩阵和 ROC 曲线评估模型的分类效果？

混淆矩阵、ROC 曲线都是基于测试集在模型中的真实结果和预测结果间的对比，无论测试结果还是真实结果都是取值二元分类变量，经过组合后形成 4 种情况即：

	预测为 0	预测为 1
真实为 0	真阴（TN）	假阳（FP）
真实为 1	假阴（FN）	真阳（TP）

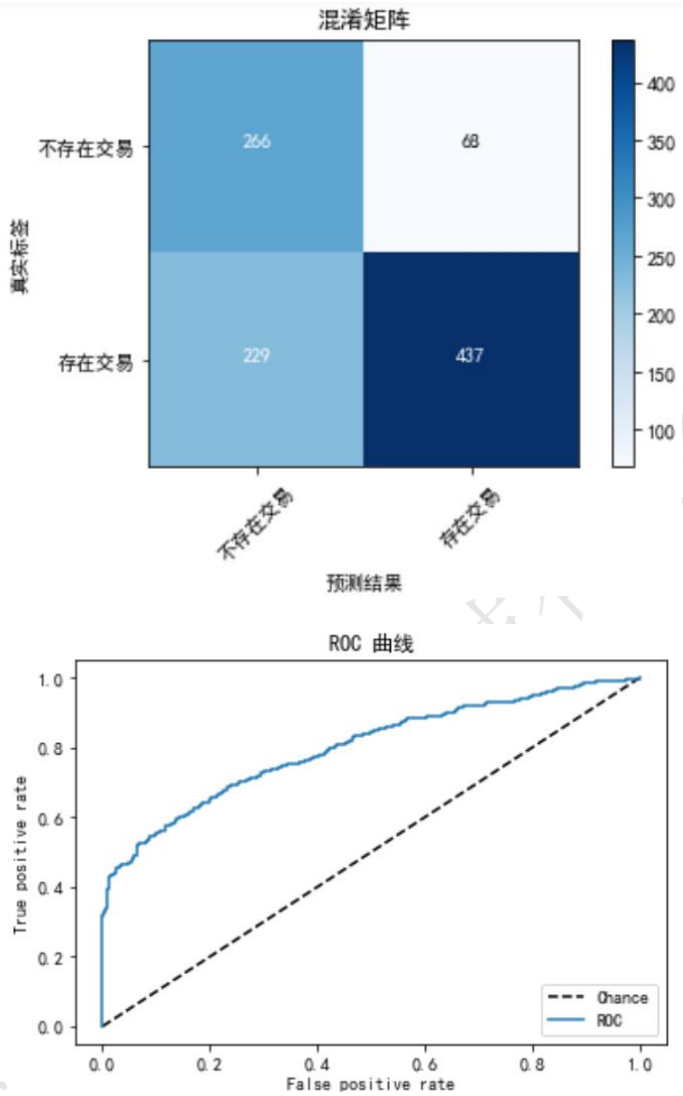
基于此可以计算以下三个模型效果评估参数：

准确度(accuracy)=(TP+TN)/(TP+FP+FN+TN)；

精确度(precision)=TP/(TP+FP)；

召回率(recall)=TP/(TP+FN)。

同时基于此可以绘制混淆矩阵图和 ROC 曲线。案例中的混淆矩阵和 ROC 曲线：



注：ROC 曲线距离左上角越近，证明分类器效果越好。

混淆矩阵的代码实现：

（1）导入并计算混淆矩阵

sklearn 提供了一个计算混淆矩阵的方法，导入代码为：

from sklearn.metrics import confusion_matrix

使用 confusion_matrix()函数计算混淆矩阵，如案例中的代码：

y_pred = clf.predict(X_test)

cnf_matrix=confusion_matrix(y_test, y_pred)

由于网格搜索超参数后得到新的最优模型，所以首先需要重新计算测试集的预测结果，其次将真实结果和预测结果放在 confusion_matrix()函数的括号中计算得到混淆矩阵，案例中将其命名为 cnf_matrix，将结果打印出来如图所示：

```
# 打印结果
print(cnf_matrix)
```

```
[[266  68]
 [229 437]]
```

（2）绘制混淆矩阵图

使用 matplotlib 中 imshow()来绘制混淆矩阵图。

将计算出来的混淆矩阵放在函数的括号中即可绘制混淆矩阵，例如案例中的代码：

plt.imshow(cnf_matrix, cmap=plt.cm.Blues)

其中 cmap 设置颜色图谱，我们使用的是 Blues，所以这个图的颜色都是蓝色调的，数值越大蓝色越深，数值越小蓝色越浅。

其次设置颜色条，X、Y 轴的刻度，文字注释和轴标签，实现一个完整的混淆矩阵图的绘制，例如案例中的代码：

```
# 制作混淆矩阵图
plt.imshow(cnf_matrix, cmap=plt.cm.Blues)
plt.title('混淆矩阵')
# 设置颜色条
plt.colorbar()

# 设置X、Y轴的刻度
tick_marks = np.arange(len(["不存在交易", "存在交易"]))
plt.xticks(tick_marks, ["不存在交易", "存在交易"], rotation=45)
plt.yticks(tick_marks, ["不存在交易", "存在交易"])

# 设置文字注释
thresh = cnf_matrix.max() / 2
for i in [0, 1]:
    for j in [0, 1]:
        plt.text(j, i, cnf_matrix[i, j],  # 文字注释的位置和内容
        horizontalalignment="center",  # 水平方向上的文字位置
        color="white" if cnf_matrix[i, j] > thresh else "black") # 文字颜色

# 设置轴标签
plt.ylabel('真实标签')
plt.xlabel('预测结果')
# 显示图像
plt.show()
```

ROC 曲线的代码实现：

（1）计算真阳率和假阳率

假阳率 FPR = FP / (FP + TN)；

真阳率 TPR = TP / (TP + FN)。

真阳率和假阳率可以通过 sklearn 中的 roc_curve()函数计算，导入该函数：

from sklearn.metrics import roc_curve

使用 predict_proba()这个方法可以获得模型中测试集的每个样本属于每一类别值的概率，例如案例中的代码：

y_score = clf.predict_proba(X_test)[:, 1]

获取经测试集 X_test 训练后的模型中，所有类别值为 1 的概率数据（在结果中 0 代表第一列，第一列内容类别值=0 的概率；1 代表第二列，第二列内容为类别值=1 的概率）

将真实结果和预测结果放在 roc_curve()函数的括号中计算真阳率和假阳率，例如案例中的代码：

fpr, tpr, thresholds = roc_curve(y_test, y_score)

得到的 fpr、tpr 分别为真阳率和假阳率，而 thresholds 表示阈值，即区间范围。

（2）绘制参考 ROC 曲线

包括绘制参考线、ROC 曲线以及设置标签和图例等。

Matplotlib 中生成图例函数为 legend()，参数 loc 可以设置图例的位置。

绘制 ROC 曲线的完整代码如图所示：

```
# 绘制参考线
plt.plot([0, 1], [0, 1], 'k--', label='Chance')
# 绘制ROC曲线
plt.plot(fpr, tpr, label='ROC')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC 曲线')
# 设置图例
plt.legend(loc='lower right')
# 显示图像
plt.show()
```

4、如何输出本案例的分析结果并对结果进行解读?

所有的数据分析都是围绕着主业务问题而进行的。

本案例的主业务问题是：

“**基于逻辑回归模型，使用公司特征预测公司是否发生交易**”

我们需要得到一个预测效果较好的逻辑回归模型和较为重要的公司特征及其系数

该模型的效果通过测试准确率和零特征占比来表示，同时筛选出案例中的非零特征和对应的系数即可。

使用 Python 中自带的 print()函数打印输出结果。

测试准确率和零特征占比：

打印函数中%.2f 是格式符，表示保留两位小数的浮点数，%%表示输出一个%，例如案例中的代码：

```
print("零特征占比 Sparsity: %.2f%%" % (np.mean(clf.coef_ == 0) * 100))

print("测试准确率 Test score: %.2f%%" % (balanced_accuracy_score(y_test, y_pred) * 100))
```

非零特征和对应的系数：

%20s 是格式符，表示输出格式是字段宽度为 20 个字符的字符串；\t 代表 Tab 符，最后使用 for 循环和 if 语句从模型中挑选出系数不为零的特征的名称和系数组合，例如案例中的代码：

```
print("以下为非零特征及对应系数：")
print("%20s\t\tcoef" % "feature_name")
for name, coef in zip(x_mapper.transformed_names_, clf.coef_[0]):
    if coef == 0:
        continue
    print("%20s\t\t%.4f" % (name, coef))
```

以及完整输出形式，如图所示：

```
零特征占比 Sparsity: 72.50%
测试准确率 Test score: 72.63%
以下为非零特征及对应系数：
        feature_name            coef
 transfer_mode_x0_做市          0.4075
        listing_days            0.5312
  registered_capital            0.2649
               asset            0.0911
           maker_num            0.8824
 net_asset_per_share            0.0128
   ave_executive_age           -0.1232
       holder_rt_1st           -0.3807
            asset_gr            0.0225
          income_gr             0.0195
  contain_org_holder            0.0242
```