内容简介

数据检视主要是通过对数据质量以及字段情况进行检视，进一步根据主业务问题和建模的需要提出数据改造方案，而在数据检视的同时，往往也会伴随着数据清理工作。

1. 主要内容

- 数据质量和字段情况；
- 数据清理；
- 提出数据改造方案。

2. 学习目标

学完本节，能解决以下问题：

- 如何将指定字段重复的数据全部筛选出来并删除错误的记录？

提出数据改造方案

在完成了数据质量和字段情况检视后，由于数据清理已经为数据分析熟于日常的工作，因此可以不再作为提出数据改造方案中的内容。除此以外，我们仍需要根据建模的需要进行必要的数据转换以及数据分割和数据重构等内容，具体如下：

1. 数据转换

通过“公司概述”的文本来分类“公司所属行业”，言下之意：

公司概述就是本案例模型的输入变量；

公司所属行业就是模型的输出变量。

两个字段都是文本型，不仅我们没有学过文本型数据如何建模，计算机也很难直接处理文本。但是基于信息论的知识，我们知道可以借助一点的数据转换方式，将文本转换为计算机可以直接处理的数字。

（1）首先对于“公司概述”这一较长文本的字段内容，通常会采用**文本向量化**的方法，文本向量化是将文本表示成一系列能够表达文本语义的向量，是文本表示的一种重要方式。

因为公司概述文本向量各维度的值表示词的权值，为了避免极值的影响，需要对各维度的取值进行归一化。

注：关于文本向量化步骤、代码实现和归一化等内容，在数据转换中会做具体介绍。

（2）其次改造公司所属行业字段，是一个定类变量，经过数据处理后，行业类型共有93种。

注：其实在数据转换中发现，部分公司概述中未涵盖任何公司所属行业，因为识别度较低再次对数据集进行了删除，随后公司所属行业类型减少了一种，因此改造的是92种行业类型。

通过查看该字段92种行业的取值情况，发现除移动互联网、电子商务、教育、院校、

学前教育等行业出现频率较高外，许多行业出现的频率都较低，可以考虑将这些出现频率较低的行业，全部归类为“其他行业”。

本案例在行业分类归类后，共产生 4 个行业标签：

[“互联网”,”电子商务”,”教育”,”其他”]

最后用数字代替行业文本。

2. 数据分割

我们设置测试集的比例为 0.2，剩下训练集和验证集的划分，通过交叉验证来实现。

同时本次案例会对：随机分割、按照分类标签分割，两种数据分割方法对模型的影响进行探讨。

注：按照分类标签进行数据分割通过设置类 train_test_split 中的参数 stratify 来实现。

3. 数据重构

在改造后的公司所属行业中，样本的分布有些不均衡，需要进行必要的数据重构处理。

在本次案例中，除了简单重采样外，会增加两种数据重采样方法：SMOTE 算法、ADASYN 算法，一种亚采样方法：简单亚采样，探究不同的数据重构对模型的影响。

注：简单重采样、SMOTE 算法、ADASYN 算法分别采用 imblearn.over_sampling 中 RandomOverSampler,SMOTE,ADASYN 函数；简单亚采样采用 imblearn.under_sampling 中的 RandomUnderSampler 函数。

4. 模型筛选思路

本案例的主业务问题是：哪个分类模型在划分公司所属行业上的表现最好？

本案例共涉及三个分类算法：决策树算法、朴素贝叶斯算法和 SVM 算法；两种不同的数据分割方法；4 种不同的数据重构方法；以及超参数搜索等可变因素。

关于模型的探讨，一般可以遵循这样的思路：

（1）如果模型涉及超参数搜索，首先进行超参数的搜索；

（2）在最优的超参数组合下，探讨数据重构的最优方法；

（3）随后对数据分割进行探讨；

（4）最后对比不同模型的最优情况，选择最佳的分类模型。

数据检视总结

1、如何将指定字段重复的数据全部筛选出来并删除错误的记录？

筛选重复数据的思路：先使用 duplicated()函数查看数据重复情况，同时设置函数中参数 keep=False，将所有重复项都标记为 True，最终将该标记放在表中，即可查看所有重复项的数据。

例如案例中在整行重复删除后，筛选公司概述重复的数据：

```
# 筛选公司概述重复的数据
industry_overview[industry_overview.duplicated('company_overview',keep=False)]
```

	company_industry	company_overview
82	运营商/增值服务	深圳市杰之龙通信技术有限公司（以下简称杰之龙公司）成立于1997年，公司主要依托于移动、电信...
357	电子商务	福州鑫煜森网络科技有限公司成立于2017年，是国内领先的游戏创客"番茄孵化器"的深度战略合作...
552	企业服务	1、我们是个综合体的独角兽公司。具有会计师事务所、代理记账公司；企业创业园孵化器（有产业园园...
993	移动互联网	福州鑫煜森网络科技有限公司成立于2017年，是国内领先的游戏创客"番茄孵化器"的深度战略合作...
1167	通信设备	深圳市杰之龙通信技术有限公司（以下简称杰之龙公司）成立于1997年，公司主要依托于移动、电信...
1364	会计/审计	1、我们是个综合体的独角兽公司。具有会计师事务所、代理记账公司；企业创业园孵化器（有产业园园...

可以发现列表索引 82 和 1167、357 和 993、552 和 1364 这三对数据 company_overview 列完全重复，但是 company_industry 列却不相同，这些数据由于存在不一致，会对模型最终的准确率产生影响，因此根据业务背景知识，人工判断 company_industry 的正确取值后，将错误的记录通过指定索引的方式删除掉，例如案例中的代码：

```
# 删除所属行业错误的记录
industry_overview.drop(index=[82,357,1364], inplace=True)
# 检查删除结果
industry_overview.describe(include='all')
```

	company_industry	company_overview
count	1201	1201
unique	93	1201
top	移动互联网	成都兴旺动物药业有限公司创建于2003年。是一家专业从事新型动物药品研发、生产和销售的高新技...
freq	236	1

删除后的样本只剩余 1201 条，且描述性统计中 company_overview 列唯一值的数量也是 1201。

知识点：

（1）duplicated()函数中参数 keep 默认取值为'first'，即除了第一次出现外剩余的重复项标记为 True，也可以设置为'last'，即除了最后一次出现外剩余的重复项标记为 True。

（2）重置索引 reset_index()函数，往往数据清理后的数据索引残缺不全，可以通过该函数重置索引，同时设置参数 drop=True 来将原索引删除掉。

内容简介

本节主要将案例中的两个文本型字段：公司所属行业、公司概述，通过一系列数据转换的操作，将其转换为建模能够使用的数据。主要包括以下内容：

1. 主要内容

- 文本向量化来处理文本型字段的步骤介绍以及代码实现；
- jieba 分词包、CountVectorizer 类、TfidfTransformer 类、TfidfVectorizer 类和 LabelEncoder 类的使用；
- 公司所属行业字段的改造。

2. 学习目标

学完本节，能解决以下问题：

- 文本向量化来处理文本型字段主要分为哪三步？
- 如何删除公司概述中不含有公司行业词语的记录？
- 如何使用 LabelEncoder()将文本转换为数值？

公司概述文本向量生成-1

公司概述字段是一整段的文字描述，这样的文本是计算机很难直接处理的，因此需要通过将本文转化为计算机可以处理的字段，一般情况下我们通过文本向量化来处理文本型字段。

1. L1、L2 范数归一化

在文本向量生成之前，首先我们了解下归一化的方法，之前我们认识到的数据归一化就是将数据统一缩放到一个特定区间，以解决不同量纲字段间的计算问题，同时提高模型的收敛速等。

这里介绍两个比较特殊的归一化方式：L1 范数归一化，L2 范数归一化。

提起 L1、L2，大家会想到 L1 正则化项（计算公式为$\sum_{i=1}^{p}|\widehat{\beta}_i|$）和 L2 正则化项（计算公式为$\sum_{i=1}^{p}\widehat{\beta}_i^2$）。在归一化的范畴内，L1 和 L2 在数学上所指与正则化是相同的，分别对应 L1 范数和 L2 范数。简单来说：

L1 范数计算的是绝对值之和，也称为曼哈顿距离；

L2 范数计算的是平方和开根号，也称为欧式距离。

在此基础上，L1 范数归一化和 L2 范数归一化就是向量的每个元素除以向量的 L1 范数或者 L2 范数。例如，L2 范数归一化的公式如下：

$$v_{norm} = \frac{v}{||v||_2} = \frac{v}{\sqrt{{v_1}^2+{v_2}^2+\cdots+{v_n}^2}}$$

2. 文本向量化的具体步骤

（1）对文本进行分词处理

为了将公司概述文本转换成向量形式，用每个词出现与否表示文本，我们首先需要对文本进行分词处理。

英文单词之间有天然分隔，因此通常按照空格分词。但是也有时候需要把多个单词作为一个分词，比如名词“New York”，需要作为一个词看待。而中文由于没有空格，分词就是一个需要专门去解决的问题。

这里用 python 的结巴分词包（jieba）进行分词，并使用停用词表（stop_words.txt）去掉停用词和标点符号。对于每条公司概述，生成的切词结果形如“词语 1，词语 2，…，词语 n”。

（2）将每段文本的分词结果向量化

这里采用词袋模型(Bag of Words,简称 BoW)，词袋模型产生的向量不考虑文本中词与词之间的上下文关系，仅仅将文本类比为一个袋子里面的所有词语的集合。

在词袋模型下，文本用一个高维向量表示，向量的每一维对应语料库中的一个词语，文本在这一维上的权重通过这个词语在文本中的词频计算。

文本向量化的形式化定义如下：

对于整个文本集合 D 中的任一文本 $d_j \in D$，可以把它表示为以下 t 维向量的形式：

$$d_j = (W_{1j}, W_{2j}, \ldots\ldots, W_{tj})$$

其中：

向量分量 W_{ij} 表示第 i 个概念 k_i 在文档 d_j 中所具有的权值；

t 为概念集中概念的个数。

在词袋模型中，权值 W_{ij} 即为概念 k_i 在文本 d_j 中出现的频次。

在本次案例中，基于分词后的语料库中共包含 14650 个中文词语，因此，每条公司概述文本都被表示成了 14650 维的向量。

注：本案例分词结果向量化是通过 scikit-learn 的 CountVectorizer 类来完成。

（3）使用 TF-IDF 进行特征的权重修正

TF-IDF（Term Frequency - Inverse Document Frequency）是指“词频-逆文本频率”，其基本思想如下：

TF 也称为局部权值，即第 i 个词语在第 j 篇文本中的权值，记为 f_{ij}。

假设 $freq_{ij}$ 为词语 k_i 在文本 d_j 中的出现次数，$maxtf_j$ 表示文本 d_j 中所有词语出现频次的最大值，则：

$$f_{ij} = freq_{ij} / maxtf_j$$

IDF 也称为全局权值，即第 i 个词语在整个文本集合 D 中的权值，记为 idf_i。

假设 N 为系统文本总数，n_i 为系统中含有词语 k_i 的文本数，则

$$idf_i = \log (N / n_i)$$

基于 TF 和 IDF 的计算，文本 $d_j = (W_{1j}, W_{2j}, ..., W_{tj})$中的每一维的权值：

$$W_{ij} = f_{ij} * idf_i。$$

局部权值 TF 体现的是词语在文本中出现的相对频率，局部权值越大，相对频率越高。

全局权值 IDF 体现的是词语对文本内容的代表性程度，n_i 越小，说明这一词语主要在这一文本中出现，全局权值越大。

W_{ij} 则综合了词语的局部权值和全局权值。

通过以上的方式，TF-IDF 可以更加合理地将文本转化为向量，并尽量使得每个词语的权值反映其真实的对文本的贡献。

经过 TF-IDF 处理后的文本向量依然是 14650 维，只是每一维的权重计算发生了变化。

注：TF-IDF 是一种加权技术，计算词频和逆文本频率之后需要进行归一化。
调用 scikit-learn 的 TfidfTransformer 类时可以通过设置参数 norm 来指定归一化的方式，默认使用 L2 范数归一化。

数据转换总结

1、文本向量化来处理文本型字段主要分为哪三步？

文本向量化来处理文本型字段主要分为以下三步：

（1）对文本进行分词处理

中文不像英文具有天然的分词分割，因此使用 python 的结巴分词包（jieba）进行分词，并使用停用词表（stop_words.txt）去掉停用词和标点符号。

在本案例中，对于每条公司概述，生成的切词结果形如：

“词语 1 词语 2 … 词语 n”

首先需要导入 jieba 包和 stop_words 停用词表，代码即停用词表如下图所示：

```
# 导入结巴分词
import jieba
# 导入停用词表
stop_words = [line.strip() for line in open('stop_words.txt',encoding='UTF-8').readlines()]
stop_words
```

```
['\ufeff',
 '!',
 '"',
 '#',
 '$',
 '%',
 '&',
 '""',
 '(',
 ')',
 '*',
 '+',
 ',',
 '-',
 '--',
 '.',
```

stop_words 文件是.txt 格式，使用 readlines()逐行 open()停用词汇，并使用 strip()函数移除每一行开头和结尾的空格。

其次使用 jieba 包对 company_overview 进行分词并去掉停用词，例如案例中的代码：

```
# 使用结巴分词对company_overview分词
for i in industry_overview.index:
    # 对每一行的company_overview分词
    segs = jieba.cut(industry_overview.at[i,'company_overview'])
    # 将结果保存在overview_seg
    overview_seg = ''
    for seg in segs:
        # 去停用词
        if seg not in stop_words and seg != ' ':
            overview_seg += seg + ' '
    # 创建新列保存分词结果
    industry_overview.at[i,'overview_seg'] = overview_seg.strip()
```

使用 for 循环依据索引依次对每条公司概述进行：分词-去掉停用词-保存结果，最后移除每行开头和结尾的空格。

同时在案例也对公司所属行业字段进行分词处理。

（2）分词结果向量化

分词向量化主要是使用 sklearn.feature_extraction.text 中的 CountVectorizer 方法，它的作用是可以将文本语料库转换为计数矩阵，即词频矩阵，使用方法也非常简单，例如案例中的代码：

```
# 构建词袋模型下的文本向量
from sklearn.feature_extraction.text import CountVectorizer

# 设置参数 去除英文停用词
vectorizer = CountVectorizer(stop_words='english')
# 数据转换 返回词频矩阵
fre_matrix = vectorizer.fit_transform(corpus)
# 查看词频矩阵
fre_matrix.toarray()
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

其中我们可以设置参数 *stop_words='english'* 来去除英文停用词。

当然除了可以使用 toarray()函数查看转换好的词频矩阵，也可以使用 get_feature_names()函数查看所有的分词名称。

注：一般情况下 CountVectorizer 中输入文本通过 tolist()函数转换 list 类型，当然不转换也是可以的。

（3）特征权重修正

采用 TF-IDF（“词频-逆文本频率”）进行特征权重的修改，其中 TF 是指局部权值，记为 f_{ij}：

$$f_{ij} = freq_{ij} / maxtf_j$$

IDF 是指全局权值，记为 idf_i：

$$idf_i = \log (N / n_i)$$

基于 TF 和 IDF 的计算，文本 $d_j = (W_{1j}, W_{2j}, ..., W_{tj})$中的每一维的权值：

$$W_{ij} = f_{ij} * idf_i$$

在 python 中通过 sklearn.feature_extraction.text 中的 TfidfTransformer 方法来完成。使用方法也非常简单，例如案例中的代码：

```
# 使用TfidfTransformer修正特征权重
from sklearn.feature_extraction.text import TfidfTransformer

transformer = TfidfTransformer()
tfidf = transformer.fit_transform(fre_matrix)
tfidf
```

```
<1180x14650 sparse matrix of type '<class 'numpy.float64'>'
        with 93373 stored elements in Compressed Sparse Row format>
```

对修改后的特征权重矩阵，同样可以使用 toarray()函数查看词频矩阵。

由于通常情况下，转换后的词频矩阵都会使用 TF-IDF 修改特征权重，因此（2）和（3）也直接使用 sklearn.feature_extraction.text 中的 TfidfVectorizer 方法来实现。

2、如何删除公司概述中不含有公司所属行业词语的记录？

一般情况下，公司概述中都会涉及到从事行业、经营范围或业务介绍等公司所属行业的词语。但由于所属行业中重叠交叉的情况较为普遍，因此我们判断删除公司概述中不含任何公司行业词语的记录。

首先利用公司所属行业字段的分词结果，创建集合。在对所属行业分词时，插入一个集合，然后将每行分词后的结果添加到该集合即可。例如案例中的代码：

```
# 公司行业词语集合
industry = set()

# 使用结巴分词对company_industry分词
for i in industry_overview.index:
    segs = jieba.cut(industry_overview.at[i,'company_industry'])
    industry_seg = ''
    for seg in segs:
        if seg not in stop_words and seg != ' ':
            industry_seg += seg + ' '
            industry.add(seg)   # 将词语添加到公司行业词语集合
    industry_overview.at[i,'industry_seg'] = industry_seg.strip()
```

其次使用 for 循环获取每一条公司概述记录词语集合，与上述集合进行查看交集，如果交集为 0，则在原表中删除这条记录，例如案例中的代码：

```
# 删除公司概述中不含有任何公司行业词语的记录
for i in industry_overview.index:
    # 获取每一条记录中公司概述词语集合
    overview = set(industry_overview.at[i, 'overview_seg'].split())
    # 删除没有交集的记录
    if len(overview & industry) == 0:
        industry_overview.drop(i, inplace=True)
industry_overview.shape
```

```
(1180, 4)
```

经过删除这些记录，最终剩余 1180 个记录。

3、如何使用 LabelEncoder()将文本转换为数值？

首先公司所属行业字段的取值有 92 种，类别太多。决定将除了“移动互联网”、“电子商务”、“教育” 3 个类别外的其余行业全部归类为 “其他”。

这里使用定义一个简单的函数，函数的逻辑可以简单理解为：

建立三个列表，对于每一个函数的输入元素进行判断，如果存在于哪个列表中，就返回这个列表的名称，如果都不属于，则返回 “其他”。最后通过 apply()将这个函数应用于公司所属行业一列。例如案例中的代码：

```
# 行业分类
internet = ['移动互联网']
ecommerce = ['电子商务']
edu = ['教育','院校','学前教育','其他技能培训','IT培训','外语培训']

# 行业字段改造
def label_transform(v):
    if v in internet:
        return '互联网'
    elif v in ecommerce:
        return '电子商务'
    elif v in edu:
        return '教育'
    else:
        return '其他'

industry_overview['company_industry'] = industry_overview['company_industry'].apply(lambda x: label_transform(x))
industry_overview['company_industry'].value_counts()
```

```
其他       690
互联网      232
教育       160
电子商务      98
Name: company_industry, dtype: int64
```

可以看到所属行业字段已经转换为 4 种取值，且可以发现其为不平衡数据集。

然后使用 LabelEncode()对其进行转换为数值表达。例如案例中的代码：

```
# 文本转换为数值
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
le.fit_transform(industry_overview['company_industry'])
```

```
array([3, 1, 2, ..., 0, 1, 1])
```

同时还可以通过 classes_查看[0, 1, 2, 3]分别代表什么：

```
# 行业类别标签
le.classes_
```

```
array(['互联网', '其他', '教育', '电子商务'], dtype=object)
```

当然如果思路清晰的话，直接在定义函数时 return 对应的数值，也是可以的。

数据分割方法

在较为严谨的数据分割中，一般涉及训练集、验证集和测试集三个数据集。

一般情况我们通过直接设置 train_test_split()函数中的 test_size 参数取值，将训练集和测试集分割为指定大小的两份，然后使用交叉验证将训练集划分为验证集和训练集。

注：参数 test_size=可以直接设置测试集的个数，还可以直接设置测试集所占的比例。（如 0.2 表示测试集占数据集的 20%）

除了设置测试集的大小，我们还可以在数据分割时设置不同的分割方式：随机分割、按照分类标签分割。默认情况下采用随机分割的方式。

按照分类标签分割是指将数据集按标签分为不同的层，在每层里按照预先设定的比例划分训练样本和测试样本，然后再将不同层的训练样本和测试样本组合形成训练集和测试集。

一般来说，按照分类标签进行数据分割更适合类分布不平衡的情况。

注：是否按照分类标签进行数据分割通过设置类 train_test_split 中的参数 stratify 来实现。

两种方法在本案例中都会有所涉及。

数据重构方法

数据重构主要的目标是将数据分布重构使其符合理想情况，主要需要解决的是数据不平衡的问题，数据重构又分为重采样和亚采样。

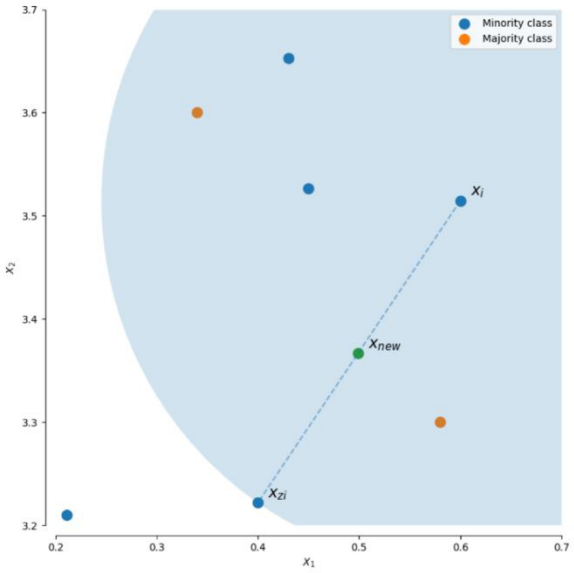
1. 重采样

在之前课程中介绍的重采样方法是简单地复制少数类样本，形成与多数类样本相同的数据集。这种方法有一个缺点，就是如果小类的样本过少，经过重采样后扩大的样本集所包含的信息依然和原样本集相同，有可能导致过拟合。

为了避免过拟合情况，我们采用在少数类中加入随机噪声、干扰数据或者通过一定规则产生新的合成样本，从而在样本集中加入新的信息，以期提高模型的泛化能力。

比较常用的改进重采样方法包括 SMOTE 算法和 ADASYN 算法。

两种算法的基本思路：基于原有样本合成新样本。如图所示：



蓝色点代表小类样本，橙色点代表大类样本，绿色点代表新的合成样本

①确定一个小类样本 x_i；

②然后在 x_i 的附近找另一个小类样本 x_{zi}；

③在 x_i 和 x_{zi} 的连线上随机选一点 x_{new} 作为新合成的小类样本。

两种算法的不同之处在于确定 x_i的方式：

SMOTE 算法随机确定 $x_{i;}$

ADASYN 算法倾向于选择那些在不同类别“边界”附近的小类样本来生成新样本。

注：两个算法的详解可参考链接 https://imbalanced-learn.org/en/stable/over_sampling.html

2. 亚采样

亚采样是对大类进行随机的采样，形成与少数类样本相同的数据集。同时这种随机的减少大类样本也会丢失大类中的一些重要信息。

注:sklearn 中提供了 RandomUnderSample 方法，可以实现简单亚采样。

数据分割与重构总结

1、什么是按照分类标签分割方法，代码如何实现？

按照分类标签分割是指将数据集按标签分为不同的层，在每层里按照预先设定的比例划分训练样本和测试样本，然后再将不同层的训练样本和测试样本组合形成训练集和测试集。

具体是否按照分类标签进行数据分割是通过设置 train_test_split()中的参数 stratify 来实现，例如案例中代码：

```
from sklearn.model_selection import train_test_split

# 分割训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.1, shuffle=True, random_state=0, stratify=y)
```

stratify 默认取值是 None，可以将其改为标签‘y’来实现按照分类标签分割数据。

2、关于新引入的这三种数据重构方法原理以及代码是什么？

三种数据重构的方法分别是：RandomUnderSample、SMOTE 算法和 ADASYN 算法，RandomUnderSample 就是之前涉及过的亚采样；而 SMOTE 算法和 ADASYN 算法都是为了避免过拟合情况，在少数类中通过一定规则产生新的合成样本，从而在样本集中加入新的信息，以期提高模型的泛化能力的两种算法。

两种算法的基本思路都是在确定一个小类样本 x_i，然后在 x_i 的附近找另一个小类样本 x_{zi}，在 x_i 和 x_{zi} 的连线上随机选一点 x_{new} 作为新合成的小类样本。

两种算法的不同之处在于 SMOTE 算法随机确定 xi；而 ADASYN 算法倾向于选择那些在不同类别“边界”附近的小类样本来生成新样本。

三种数据重构的方法在代码实现上也非常容易，如下所示：

```
from imblearn.over_sampling import RandomOverSampler,SMOTE,ADASYN
from imblearn.under_sampling import RandomUnderSampler

# 简单重采样
ros = RandomOverSampler(sampling_strategy='auto', random_state=0)
X_train_ros, y_train_ros = ros.fit_resample(X_train, y_train)
# SMOTE重采样
sm = SMOTE(random_state=0)
X_train_sm, y_train_sm = sm.fit_resample(X_train, y_train)
# ADASYN重采样
ada = ADASYN(random_state=0)
X_train_ada, y_train_ada = ada.fit_resample(X_train, y_train)
# 简单亚采样
rus = RandomUnderSampler(random_state=0)
X_train_rus, y_train_rus = rus.fit_resample(X_train, y_train)
```