目录

1 数据预处理概述

1.1 数据预处理的必要性

欧盟机构于1999年联合起草的跨行业数据挖掘标准流程(Cross-Industry Standard Process for Data Mining，CRISP-DM) 包含6个步骤: 业务理解、数据理解、数据准备、建立模型、模型评价和模型实施(如图1.1所示). 其中的“数据准备”, 主要内容是数据预处理[1]. CRISP-DM说明: 在数据分析中, 应该首先了解业务、厘清分析目标和思路; 其次需依据分析目标采集适当的数据并充分地理解数据; 随后进行数据预处理, 最后才正式开展具体的分析. 可见, 数据预处理是整个数据分析流程中重要的一环, 它连接了数据输入与后续的数据分析.

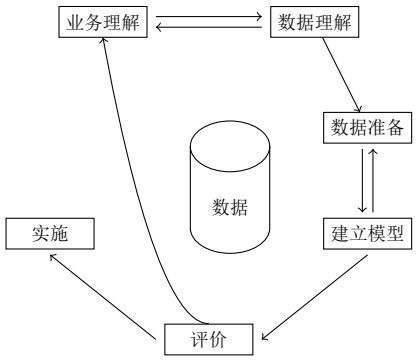


图 1.1: 跨行业数据挖掘标准流程示意图

[1]本书中数据预处理是指在建立模型以前所进行的数据处理.

此外, 数据分析领域从业者几乎都认同以下三种说法:

(1) 业务第一, 数据次之, 算法第三.

(2) 好的数据胜于好的特征, 好的特征胜于好的算法.

(3) 数据和特征决定了机器学习的上限, 而模型和算法只是逼近这个上限而已.

其实, 这三种说法含义相近, 都告诉我们: 高质量的数据和适应分析目标的特征或变量, 是建立模型的重要前提和基础, 这也成为了数据预处理的目标. 对原始数据进行预处理, 不仅可以保证后续数据分析顺利进行, 还能改进数据的质量, 从而提高数据分析的效率和精度.

但是, 数据分析工作者应当遵从和坚持数据的真实性, 而数据预处理往往涉及对数据的修改, 因而需要十分谨慎地把握数据预处理的尺度. 一般, 我们只需作必要的预处理, 宁简勿繁、稳中求进, 不倡导过度的预处理. 那么, 哪些是必要的数据预处理呢? 我们认为, 必要的数据预处理应至少符合下列情况中的一种.

(1) 数据预处理保证数据分析得以实现. 某些预处理一旦未被执行, 就无法实现后续的数据分析, 那么这类数据预处理就是必要的. 例如, 当采用CART算法建立决策树模型时, 若采用Python语言scikit-learn库中的tree模块, 由于它不允许缺失值, 也不能直接处理定性变量, 因此必须对缺失值和定性变量进行处理, 否则构建模型的代码无法运行.

(2) 数据预处理为正确的数据分析保驾护航. 一方面, 许多统计学模型基于一定的假设发展而来. 例如, 在普通最小二乘回归(OLS) 模型中, 若我们使用检验的P值和区间估计等结果, 则要求模型的误差项服从正态分布, 若不满足要求, 结果可能不可靠. 当误差项的正态性假定不成立时, 在某些情况下我们可以通过数据预处理, 例如采用Box-Cox变换, 使得误差项服从或近似服从正态分布. 另一方面, 一些模型方法自身的特点(如: 对变量尺度敏感, 易受离群值影响等) 决定了数据须满足一定的要求. 例如, 基于明氏距离(Minkowski distance) 寻找邻居的k近邻方法, 它对变量的尺度敏感, 并且通常不能直接应用于定性变量. 此时, 需要对定量变量作规范化处理, 并对定性变量作独热编码, 这样才能保证所建立的模型准确可靠.

(3) 数据预处理可满足高效开展数据分析的需要. 例如, 在支持向量机和神经网络模型中, 需要使用梯度下降方法, 通过多次迭代获得模型参数的估计值. 为了对所有参数作统一的初始值设置和迭代设置, 一般需要对

定量变量作规范化处理、对定性变量作独热编码, 从而保证高效地实现参数估计.

在实际操作中, 应当结合以下几个因素判断哪些数据预处理是必要的:

(1) 分析目标. 实际上, 分析目标决定了整个数据分析过程, 包括数据预处理. 例如, 只有当分析目标是预测类别时, 才需要考虑是否存在类别不平衡问题, 以及是否需要进行类别不平衡处理.

(2) 数据特点. 统计学的特点是"分而治之", 尤为重要的一点是根据数据的特点(例如, 变量类型, 样本观测之间的相依关系等) 采用不同的数据分析和处理方法. 例如, 当数据中含有缺失值时, 才需要考虑处理缺失值.

(3) 分析方法. 不同的数据分析方法对数据质量要求不同, 因此数据预处理应适应分析方法的需求. 例如, 对于自变量中的离群值, 决策树模型一般并不敏感, 因而可以不必处理离群值; 而普通最小二乘回归模型则对离群值较为敏感, 可能需要在建模前处理离群值.

(4) 软件. 不同的软件和软件包都有特定的设置, 它们对数据的要求不尽相同, 因而对应的数据预处理也会有差异. 作为一个例子, 表1.1对比了采用不同的软件包建立CART模型, 对数据预处理的不同要求. 例如, R语言rpart包中建立CART模型的函数可以直接处理缺失值和定性变量(取值为数值或字符串均可), 此时, 缺失值处理和独热编码就不再是必要的.

当然, 所谓的"必要"或"不必要"并不一定有明显的界限, 那些可以提升模型的可解释性和泛化能力等的预处理都是可以考虑的. 表1.2展示了当使用Python语言scikit-learn库时, 一些常用的模型方法所需要的部分数据预处理. 其中, 标识为"必要", 表明数据预处理是必要的; 标识为"可选用", 表明数据预处理虽不一定必要, 但具有提升模型效果的功能. 例如, 决策树方法一般无法适应类别不平衡问题, 需要通过预处理形成类别平衡的数据, 从而提升模型的效果. 需要注意的是, 表1.2仅展示了部分数据预处理, 并未完整地涵盖模型对数据预处理的要求. 例如, LightGBM不能接受取值为字符串的定性变量, 它虽不要求对定性变量作独热编码, 但需要作数值编码, 将字符串替换为数值. 由此可见, 是否进行数据预处理、采用哪些预处理方法, 需要具体情况具体分析.

1.2 数据预处理的步骤

数据预处理大致可分为4个方面:

表 1.1: 构建CART模型所需要的部分预处理

软件/软件包	缺失值处理	定性变量独热编码	定量变量规范化	离群值处理	特征选择与提取	类别不平衡处理
Python: scikit-learn库	必要	必要	-	-	-	可选用
R语言: rpart包	-	-	-	-	-	可选用

表 1.2: 实现scikit-learn中的模型所需要的部分预处理

模型	缺失值处理	定性变量独热编码	定量变量规范化	离群值处理	特征选择与提取	类别不平衡处理
OLS/Logistic回归	必要	必要	-	可选用	可选用	可选用
决策树/随机森林	必要	必要	-	-	-	可选用
XGBooost	-	必要	-	-	-	可选用
LightGBM	-	-	-	-	-	可选用
支持向量机	必要	必要	必要	-	可选用	-
k近邻	必要	必要	必要	-	可选用	-
神经网络	必要	必要	必要	可选用	-	可选用

(1) 数据集成. 包含特征衍生和类别不平衡问题处理等.

(2) 数据清理. 包含清洗脏数据、离群值处理和缺失值处理等.

(3) 数据变换. 包含规范化和离散化等.

(4) 数据归约. 包含数值归约、特征选择和特征提取等.

可以将特征衍生、特征选择和特征提取合并, 称为"特征工程".

现实的数据预处理没有固定的内容和方法. 对同样的一组数据, 两位数据分析工作者可以选择不同的数据预处理内容和方法, 这可能导致数据分析结论截然不同. 我们常说统计学兼具科学性和艺术性. 可以看到, 数据预处理的科学性和艺术性尤为明显. 在一个完整的数据分析过程中, 数据预处理一般要花费约70%至90%的时间[2], 可能需要通过数个步骤才能完成. 但是, 要形成具有普适性的数据预处理步骤是困难的. 一方面, 对数据预处理的需求依赖于多种因素, 例如1.1节提到的分析目标、数据特点、所选用的分析方法和软件等. 这告诉我们, 不存在一招制胜的法宝, 必须根据具体问题和数据, 设计数据预处理方案和步骤. 另一方面, 从图1.1中数据准备与建立模型之间的双向箭头可以看出, 数据预处理需要根据模型构建过程和结果交互地、反复地进行. 这也决定了数据预处理往往是动态的, 需要在探索和尝试中不断调整, 很难一步到位. 由此可见, 不存在固定的数据预处理顺序和步骤, 可以根据需要予以安排. 当然, 在一个具体的数据分析项目中, 可以结合数据分析的效果, 评价和优化数据预处理方案.

例1.1. 对于一个分类问题(因变量为定性变量), 考虑采用支持向量机模型. 注意到支持向量机模型对离群值并不太敏感, 并且有一定的处理类别不平衡问题的能力, 因此预处理中不需要进行离群值处理和类别不平衡处理. 考虑到支持向量机对变量的尺度敏感, 因此需要对定量变量作规范化处理. 此外, 支持向量机不能自动学习和筛选特征, 开展一定的特征工程有助于提升模型效果. 若采用Python语言scikit-learn库中SVM函数建立模型, 需要注意它不允许缺失值, 并且不能直接处理定性变量. 由此, 一种可能的数据预处理内容和步骤安排如下:

步骤1: 清洗脏数据;

步骤2: 采样;

步骤3: 缺失值处理;

[2]相关调查参见https://whatsthebigdata.com/2016/05/01/data-scientists-spend-most-of-their-time-cleaning-data/.

步骤4: 特征衍生;

步骤5: 规范化(包含对定性变量作独热编码和对定量变量规范化);

步骤6: 特征选择;

步骤7: 特征提取.

上述步骤的安排, 基于以下几点考虑:

(1) 通过清洗脏数据, 将明显错误和不合乎规范等有问题的数据进行处理, 以保证后续的数据预处理正常进行.

(2) 通过采样, 将数据集划分为训练集和测试集. 许多数据预处理方法涉及参数估计或模型训练, 应基于训练集进行数据预处理的训练. 当样本量或变量维数特别大时, 也可以通过采样获得数据子集, 以提升数据分析的可行性与效率. 无论何种情况, 采样一般在数据预处理初期进行.

(3) scikit-learn中许多数据预处理的函数都不允许缺失值, 因此缺失值处理成为其他诸多预处理的前提.

(4) 特征衍生靠前放. 一方面, 衍生变量可能涉及后续的预处理; 另一方面, 衍生变量的有效性尚待考量, 可能需要通过后续的特征选择等予以修正.

(5) 某些特征选择方法涉及建模, 需要在此之前对变量规范化.

(6) 待特征选择后再实施特征提取, 有利于提高特征提取的效率.

请注意, 例1.1中数据预处理内容和步骤的安排只是数种可能方案中的一种. 并且, 其中的某些预处理可能需要反复进行, 例如, 可能需要对特征提取所产生的变量进行筛选, 此时需从步骤7跳转回步骤6.

例1.1中未考虑离群值和类别不平衡处理. 如果需要, 可以将离群值和类别不平衡处理放置于特征选择之前, 这是因为某些特征选择方法涉及建模, 而模方法往往要求先处理离群值和类别不平衡问题.

最后, 给出关于数据预处理的几点提示:

(1) 对于看起来非常完美的数据, 我们也要相当谨慎, 此时仍可能需要作数据预处理. 例如, 一些数据库系统默认以0代替空缺值, 当面对这类看似完整的数据时, 我们应当详细地了解数据采集和记录的机制, 分析潜在的问题, 并开展相应的数据预处理.

(2) 根据具体情况设计与安排数据预处理. 需要强调的是, “具体”应该是非常细节的. 例如, 若需要对普通最小二乘回归模型作检验或获得参数的区间估计, 则需要误差项满足一定的假设, 此时, 为了满足假设而进行的数据预处理就是必要的. 但若仅使用普通最小二乘回归模型中的点估计值, 模型的假设则无足轻重, 为了满足假设而进行的数据预处理就不再是必要的. 因此, 对整个数据分析细节的、整体性的考虑非常重要, 在此基础上进行的数据预处理才可能高效.

(3) 应注意数据预处理的可行性. 例如, 当插补缺失值时, 应避免使用模型中的因变量. 这是因为当模型部署到实际业务中时, 因变量的值往往是未知的, 无法应用于数据预处理. 这种情况下, 含有因变量的预处理是不可行的.

(4) 数据预处理是一个动态的过程, 往往需要不断尝试、循环往复, 很难一步到位. 掌握好每一个预处理方法, 在实践中多作尝试, 灵活运用, 这些都是必要的.

1.3 数据预处理评价与优化

数据预处理是整个数据分析的一部分, 对数据预处理的评价应基于分析目标的达成程度. 在图1.1所示的数据挖掘流程图中, 评价发生在数据预处理与建立模型之后, 是对数据预处理和建立模型的整体评价. 因此, 通常的模型评价方法即可用于数据预处理的评价.

为了客观地评价数据预处理和模型效果, 最常用的方法是留出法(holdout method), 即将数据集划分为训练数据(training data) 和测试数据(test data), 在训练数据中进行预处理并建立模型(包含调参优化), 并基于测试数据评价效果. 也可以将数据集的划分作为数据预处理的一个部分. 当样本观测不具有时间顺序时, 可以通过随机采样划分数据集; 当样本观测具有时间顺序时, 一般应将时间在前的样本观测放入训练集, 时间在后的样本观测放入测试集, 从而与实际业务逻辑相吻合.

注记1.1. 留出法的核心是避免数据泄露(data leak), 因此涉及到“训练”的数据预处理都应基于训练集开展, 这包含涉及参数估计的数据预处理方法(如: 部分规范化、离散化和缺失值插补等), 以及需要建立模型的数据预处理方法(如: 部分类别不平衡问题处理、缺失值插补、特征选择和特征提取等). 因此, 对数据集的划分应在数据预处理的初始阶段.

训练集一般约占数据集的70%-90%. 相应地, 用于模型效果评价的测试集约占数据集的10%-30%. 为了调参优化(包含对数据预处理的优化和对

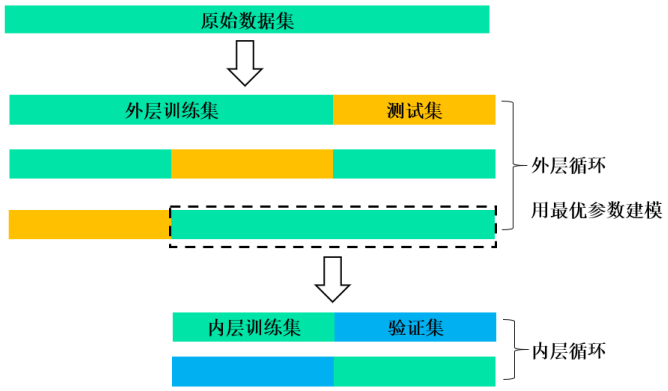


Figure 1.2: 外层为3折、内层为2折的嵌套交叉验证示意图

后续模型的优化), 还可以进一步将训练集划分为建模训练集和建模验证集(validation data set), 它们各约占整个数据集的50%-70%和10%-30%. 为提升数据使用效率, 也可采用交叉验证(Cross Validation, CV), 使得划分的几个部分数据集可轮流参与建模或验证, 以提升数据信息的使用效率.

但是, 若仅实施一次留出法, 评价的结果带有偶然性. 建议采用重复留出法(repeated holdout method), 即多次划分训练集和测试集, 每一次划分下都可以产生效果评价, 最终将各次评价结果平均. 嵌套交叉验证(nested cross-validation) 是重复留出法的一个特例, 它采用交叉验证的方式轮流选用一个数据集作为测试集, 同时在训练集内部实施交叉验证进行调参优化, 因此嵌套了两层交叉验证. 图1.3展示了一个嵌套交叉验证示例: 内层为用于参数优化的2折交叉验证, 外层是作为重复留出的3折交叉验证.

在数据分析实践中, 我们可以将数据预处理的训练与模型的训练结合在一起, 将二者合并为一个整体, 称为管道(pipeline). 这不仅便于统一处理, 还可有效避免数据泄露.

1.4 案例分析

R语言rpart包中的car.test.frame数据集(以下简称为car数据集) 是取自于1990年4月《消费者报告》的部分汽车品牌数据, 含有60个样本观测和8个变量(见表1.3). 欲以Mileage为因变量, 其他变量为自变量, 建立预测模型. 下面读入数据, 并设置自变量和因变量. 其中, 自变量Reliability应为定序变量, 但原始数据为浮点型, 会被作为定量变量处理, 因此将其类型改为'object'.

由于因变量是定量变量, 因此选用决定系数R^2作为效果的评价指标, R^2越大表明数据预处理与模型的效果越好. 考虑两种模型方法: CART模型

表 1.3: car数据集变量信息

序号	变量名称	变量含义	变量类型	取值(范围) 及其含义
1	Country	原产地	定类	'USA', 'Korea', 'Japan/USA', 'Japan', 'Mexico', 'Germany', 'France', 'Sweden'
2	Reliability	可靠性	定序	1,2,3,4,5
3	Price	价格	定量	5866-24760 (美元)
4	Type	类型	定类	'Small', 'Sporty', 'Compact', 'Medium', 'Large', 'Van'
5	Weight	车辆自重	定量	1845-3855 (磅)
6	Disp.	排量	定量	73-305 (升)
7	HP	净马力	定量	63-225
8	Mileage	油耗(每加仑燃油可行驶英里数)	定量	18-37 (英里)

和随机森林模型, 希望从中择优选择一个模型. 采用scikit-learn中DecisionTreeRegressor函数建立CART模型, 以网格搜索的方式优化参数: 最小叶节点样本量min_samples_leaf, 备选值为3和5. 采用RandomForestRegressor函数建立随机森林模型, 以网格搜索的方式优化3个参数, 每一参数考虑2个备选值: 节点中抽取的备选分枝变量最大比例max_features为0.5和0.7, 最小叶节点样本量min_samples_leaf为3和5, 个体学习器的数量n_estimators为20和50. 其余参数采用函数中的默认值.

由表1.3看到, 自变量中包含三个定性变量, 其中Country和Type的取值为字符串, Reliability的取值为数值. 由探索性分析发现, 仅Reliability含有缺失值, 涉及11个样本观测, 约占18%. DecisionTreeRegressor函数与RandomForestRegressor函数都不能直接处理定性变量, 也不允许缺失值. 因此, 对定性变量规范化和处理缺失值都是必要的预处理. 具体而言, 根据两个模型函数的要求, 需要对定性变量作独热编码. 这里的缺失值发生在定性变量之中, 可以有多种处理方式, 这里考虑两种简单的方式:

(1) 以众数插补缺失值.

(2) 将缺失值作为新的类别值, 在独热编码中增加一个虚拟变量.

下面, 评价和选择缺失值处理方式.

第一种数据预处理方案: 以众数插补定性变量的缺失值, 并对定性变量作独热编码. 同时, 以中位数插补定量变量的缺失值. 本例中的定量变量实际上没有缺失值, 但完整地考虑两类变量的缺失值插补是一种好习惯, 以应对应用中可能出现的各种缺失数据情况.

```
categorical_features = np.where(X.dtypes == object)[0]
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent',
       fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])

numeric_features = np.where(X.dtypes != object)[0]
numeric_transformer = Pipeline(steps=[('imputer',
    SimpleImputer(strategy='median'))])

preprocessor1 = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
           ])
```

第二种数据预处理方案: 仅对定性变量独热编码, 让缺失值成为一个虚拟变量. 代码与第一种数据预处理方案类似, 只需要将categorical_transformer替换为categorical_transformer2, 其代码为:

```
categorical_transformer2 = Pipeline(steps=[('onehot',
    OneHotEncoder(handle_unknown='ignore'))])
```

最理想的方式是将数据预处理优化与模型优化整体对待. 具体而言, 需要探索数据预处理方式与模型的所有组合之中, 哪一种组合更优. 同时, 考虑到数据预处理需要基于训练集进行训练[3], 将数据预处理与模型构建组合为一个管道进行训练是合理的. 本例中有两种数据预处理方式和两种模型, 因此可组合为四个管道.

对于每一管道, 可采用嵌套交叉验证进行优化: 内层为用于优化的2折交叉验证, 外层是作为重复留出的3折交叉验证(如图1.3所示). 假定在外层交叉验证中, 将数据集随机划分为3个部分, 分别记为A_1, A_2, A_3. 例如, 将A_1作为测试集, A_2, A_3作为训练集, 则将$A_2 \bigcup A_3$随机划分为2个部分B_1和B_2. 在每一组备选参数组合下, 需要以B_1为训练集作预处理并建立模型, 基于B_2评价结果. 然后, B_1与B_2交换角色, 类似地获得每一组备选参数组合的训练和评价结果. 将每一参数组合下的2项评价结果平均并作比较, 可以找到最优的参数组合. 采用最优的参数组合, 基于A_2, A_3训练模型, 并由A_1评价模型(结果如表1.4所示). 将三次留出法的结果平均并求标准差, 它们显示了在所设置的数据预处理方案和模型的待优化参数组合情况下, 基于数据集中2/3的数据量进行预处理并建模所能实现的预测效果.

以第一种数据预处理方案+CART模型为例, 实现嵌套交叉验证.

```
grid_min_samples_leaf = [3,5]
param_grid_tree = [{'tree__min_samples_leaf':
    grid_min_samples_leaf}]
tr = DecisionTreeRegressor(random_state=1)
tree_pipe1 = Pipeline([
   ('preprocessing', preprocessor1),
   ('tree', tr)])
```

[3]本例中, 独热编码依赖于训练集, 因为独热编码所产生的虚拟变量数量依赖于训练集中定性变量的取值. 同时, 用于插补缺失值的众数需要在训练集中拟合而得.

```
gs_tree1 = GridSearchCV(estimator=tree_pipe1,
              param_grid=param_grid_tree,
              scoring='r2',
              cv=KFold(n_splits=2, random_state=1, shuffle=True
                  ))

scores_tree1 = cross_val_score(gs_tree1, X, y, scoring='r2',
    cv=KFold(n_splits=3, random_state=1, shuffle=True))

print(scores_tree1)
print((np.mean(scores_tree1),np.std(scores_tree1)))
```

对于随机森林模型, 只需要更改相应的参数网格设置和模型设置.

```
grid_max_features = [0.5, 0.7]
grid_min_samples_leaf = [3, 5]
grid_n_estimators = [20, 50]
param_grid_rf = [{'rf__max_features':grid_max_features,
                'rf__min_samples_leaf':grid_min_samples_leaf,
                 'rf__n_estimators':grid_n_estimators}]

rf = RandomForestRegressor(random_state=1)
rf_pipe1 = Pipeline([
   ('preprocessing', preprocessor1),
   ('rf', rf)])
```

由表1.4可以看出:

(1) 在任一种模型方法下, 两种数据预处理方式下的R^2差异在1个标准差范围内. 因此, 可以认为两种数据预处理方式对模型效果没有显著影响. 相对而言, 将缺失值作为新类别的处理方式所得到的R^2的标准差更小, 表明这种处理方式受数据随机划分的影响更小.

(2) 相对于CART模型, 随机森林模型R^2平均值更大, 且标准差更小. 因此, 在所设置的参数组合情况下, 随机森林模型比CART模型更优.

总结而言, 将缺失值作为新类别的处理方式与随机森林模型的组合为最优.

选定数据预处理方式和模型方法后, 可以进一步细化模型参数的搜索网格, 基于交叉验证(每一折交叉验证中仍然需要训练数据预处理和模型, 并基

表 1.4: 嵌套交叉验证中外层的R^2及其平均值和标准差

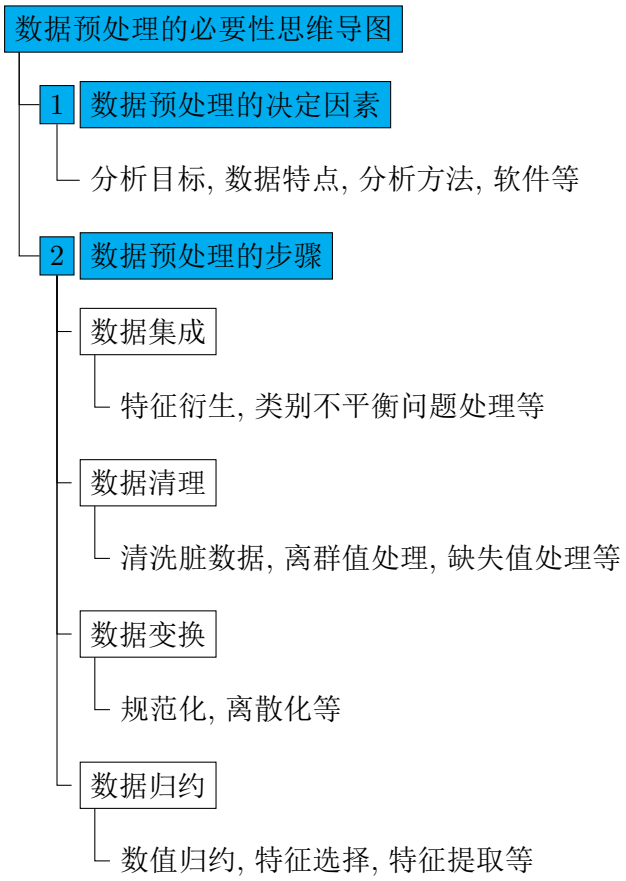
外层	众数插补 +CART	缺失值为新类别 +CART	众数插补 +随机森林	缺失值为新类别 +随机森林
A_2&A_3为训练集, A_1为测试集	0.769	0.769	0.824	0.757
A_1&A_3为训练集, A_2为测试集	0.718	0.718	0.730	0.671
A_1&A_2为训练集, A_3为测试集	0.506	0.550	0.610	0.740
平均值	0.664	0.679	0.721	0.723
标准差	0.114	0.094	0.087	0.037

于验证集评价效果) 得到最优的参数组合. 最后, 采用最优的参数组合, 基于全体数据集训练得到最终的随机森林模型, 其效果可以基于袋外误差[4]评判或待模型部署上线后在未来业务中获得.

[4]随机森林可以采用Boostrap采样建立个体学习器, 未入样的样本观测为袋外(out-of-bag)样本, 它们可作为测试集, 评价个体学习器的效果, 在此基础上获得袋外误差.

本章小结

(1) 思维导图



(2) Python函数

表1.5列出了以网格搜索优化参数和交叉验证的Python函数.

习题

1. 请尝试列举几个你感兴趣的模型/算法在特定软件中对数据预处理的要求, 从而分析哪些是必要的数据预处理.

2. 针对天猫用户重复购买预测案例, 根据跨行业数据挖掘标准流程设计数据分析方案, 特别是细化构思数据预处理方案. 提示: 该案例来源于阿里云天池的学习赛https://tianchi.aliyun.com/competition/entrance/231576/introduction.

表 1.5: 网格搜索优化参数和交叉验证Python函数

方法	函数名称与所属模块	重要参数	应用提示
建立管道	Pipeline (scikit-learn库pipeline)	-	管道中各项处理按序实施, 建模函数应放在最后
网格搜索优化参数	GridSearchCV (scikit-learn库model_selection)	(1) estimator: 所采用的模型 (2) param_grid: 需要搜索的参数及其备选值 (3) scoring: 模型效果评价指标, 默认值为None (4) cv: 交叉验证的设置. 默认值为None, 采用5折交叉验证, 其中shuffle=False, 会导致按样本观测原始排序分组, 缺乏随机性. 建议设置为shuffle=True的交叉验证 (5) refit: 重新拟合模型. 默认值为True, 表示将采用由网格搜索得到的最优参数组合基于全体数据拟合模型	需根据estimator的要求进行数据预处理
交叉验证	cross_val_score (scikit-learn库model_selection)	estimator, scoring, cv, 含义同上	需根据estimator的要求进行数据预处理

References

[1] 吴翌琳, 房祥忠(2016) 大数据探索性分析. 北京:中国人民大学出版社.