# 摘 要

在数据分析与大数据技术越发成熟的今天，越来越多的行业对数据分析的需求日益增多。论文叙述了从零搭建个人网站的整个过程，并对在搭建网站中所运用的知识以及工具都有详细的介绍和讲解。比如：用于存储数据的关系型数据库mysql；用于和数据库交互的基于python的后端django框架；用于编写前端显示页面的vue框架；用于代码管理工具以及版本控制系统的git技术；用于编写后端python代码的pycharm ide软件;用于编写前端html,css,javascript的Visual Studio Code编辑软件.

**关键词**：网站搭建， mysql, django, vue, git

# ABSTRACT

The construction of personal website can not only consolidate their own relevant knowledge, but also in the process of building a website to learn the actual network knowledge system, so as to accumulate knowledge and practical experience, for students engaged in website design work is a good opportunity to exercise.

This paper describes the whole process of building a personal website from scratch, and the knowledge and tools used in building a website are introduced and explained in detail. For example: mysql, a relational database for storing data; A python-based back-end django framework for interacting with databases; Vue framework for writing front-end display pages; Git technology for code management tools and version control systems; Pycharm ide for writing back-end python code; Used to write front-end HTML, CSS,javascript Visual Studio Code editing software

**The keywords：**Website construction, mysql, django, vue, git

# 目 录

摘 要 1

ABSTRACT 2

目 录 3

1绪论 5

1.1 数据挖掘代码生成器的背景和意义 5

1.2 自动代码生成技术的发展状况 5

1. 3 数据挖掘代码生成器的研究内容 7

1. 4 系统功能简述 8

2 数据挖掘代码生成系统分析 8

2. 1 整体架构分析 8

2.1.1 系统架构图 8

2.2 网站需求分析 9

2.2.1 需求分析概述 9

2.2.2 网站系统流程图 10

2.3 本章小结 10

3 数据库分析与设计-MongoDB 11

3.1 后台数据库以及MongoDB概述 11

3.1.1 Mongodb数据库介绍 11

3.1.2 选择MongoDB数据库分析 11

3.2 数据集合的具体设计 11

3.2.1 Mongodb数据库设计规范 11

3.2.2 数据库集合设计展示 11

3.3 本章小结 13

4 数据挖掘代码生成器模板代码设计 13

4.1 数据清洗代码生成器 13

4.1.1 常见数据清洗方法简介 13

4.1.2 数据清洗代码模板设计 14

4.1.3数据清洗代码生成交互 15

4.2 数据建模（机器学习）代码生成器 16

4.2.1 有监督与无监督学习算法 16

4.2.2 模型评估方法简介 17

4.2.3 数据建模机器学习方法优劣 19

4.2.4 数据建模（机器学习）代码生成模板设计 24

4.3 本章小结 27

5 数据分析系统引擎设计 27

5.1 数据清洗引擎设计 27

5.2 数据建模（机器学习）引擎设计 28

6 上线部署 30

参考文献 31

致 谢 32

# 1绪论

## 1.1 数据挖掘代码生成器的背景和意义

近年来，数据挖掘引起了信息产业界的极大关注，其主要原因是存在大量数据，可以广泛使用，并且迫切需要将这些数据转换成有用的信息和知识。获取的信息和知识可以广泛用于各种应用，包括商务管理、生产控制、市场分析、工程设计和科学探索等。数据挖掘利用了来自如下一些领域的思想：来自统计学的抽样、估计和假设检验；人工智能、模式识别和机器学习的[建模技术](https://baike.baidu.com/item/%E5%BB%BA%E6%A8%A1%E6%8A%80%E6%9C%AF/19041163)和学习理论。数据挖掘也迅速地接纳了来自其他领域的思想，这些领域包括最优化、进化计算、信息论、信号处理、可视化和信息检索。一些其他领域也起到重要的支撑作用。特别地，需要数据库系统提供有效的存储、索引和查询处理支持。源于高性能（并行）计算的技术在处理海量数据集方面常常是重要的。分布式技术也能帮助处理海量数据，并且当数据不能集中到一起处理时更是至关重要。

随着互联网各种的技术的成熟以及应用的落地，数据量的与日俱增，不同行业对数据分析的需求也因此产生了巨大的增长。此外，在目前的各种互联网产品中，后台存储的大量数据拥有着巨大的挖掘价值，随着机器学习技术在各领域的广泛应用，使得对大批量数据的建模分析变得容易。

然而，在企业实际运作中，由于不同数据源产生的数据往往有着不同字段，以及不同的预处理逻辑，使得面对不同的数据集时需要针对特定场景进行代码的重新编写。鉴于数据挖掘流程有着明确的处理流程，高可复用的逻辑，即数据探索、数据清洗、数据建模，模型评估等步骤。本文着眼于将数据挖掘流程模板化，通过用户在界面选择对应方法，个性化定制，生成以用户上传数据集为分析目标的python数据挖掘代码，提高数据分析流程的效率。

## 1.2 自动代码生成技术的发展状况

自动代码生成技术从20世纪末的萌芽阶段发展之今,已经趋于平稳和成熟,“自动代码生成”从1991年开始出现相关研究,2015年达到最热｡如图1所示(统计图中的纵轴代表研究成果,文中的数据统计于百度学术)

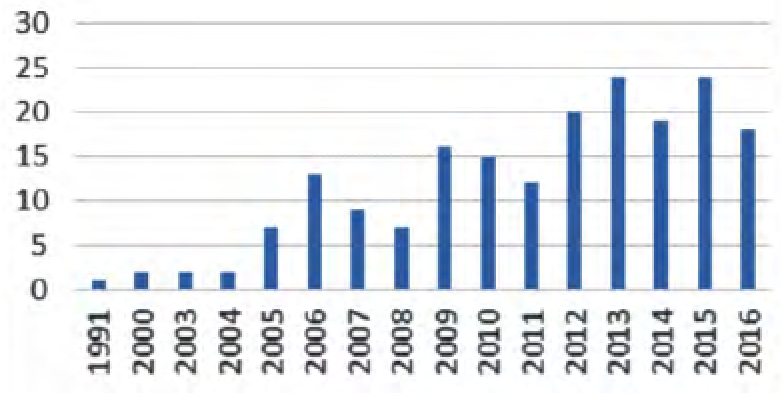


图1自动代码生成技术趋势图

自动代码生成是一个多领域多学科交叉的学科,涉及的面比较广｡随着研究的不断深入,出现了越来越多与“代码生成”相关的研究点,形成了庞大的研究网络,该网络囊括:软件开发､编译器､模型驱动､嵌入式系统､设计模式和编译程序等领域的知识｡国内的软件企业在自动代码生成和模型驱动构架的研究和开发中也有不少成果,比如金蝶软件公司是首家实现支持模型驱动构架软件开发的商业工程工具金蝶EAS4.0,是一个基于模型驱动架构､采用“业务模型驱动的全自动化软件工厂”开发的企业应用平台｡楚凡科技是专业复杂软件及系统开发解决方案的供应商｡为软件开发的整个生命周期提供集成的产品和专业化服务,楚凡科技工具KantStudio集合了UML建模和数据库建模;实现数据库模型和UML模型的双向转换;实现代码和模型之间的迭代转换｡

一些中小企业和开源产品在上自动代码生成方面也有较好表现,开发了小型的代码生成器支持从模型或者模板到不同程序设计语言的转化｡如动软.Net代码生成器是基于关系数据库元数据库的C#自动代码生成器,Codematic生成的代码,基于面向对象思想和三层架构设计｡BBOSS自动代码生成工具是一款为专有框架和平台生成前端､后端代码､WEB服务代码､SQL配置､IOC配置的可视化配置管理工具｡CodeGenerator生成工具是基于Freemarker和Velocity模板生成代码的引擎,该生成器是根据关系实体的元数据来生成源代码｡

国内的众多高校和研究机构也在代码生成､尤其是基于模型驱动骨架的代码生成软件工程也进行了长期的研究,取得了众多的成果｡下面给出国内的在自动代码生成方面研究的典型问题,文献｡在基于模型转换和代码生成､对象关系映射方面的自动代码生成,基于模式的用户界面自动代码生成技术,在逆向工程和程序流程图到自动代码生成提出一种基于图的生成算法,在平台无关模型到平台相关模型,web用户界面建模和自动生成,用户界面代码自动生成等等[1]

## 3 数据挖掘代码生成器的研究内容

本文主要研究通过python实现基于Web端的数据挖掘代码生成器。代码的生成主要采用基于基于模板的代码生成方法。基于模板的代码生成的基本原理是把软件的需求分成两部分,一部分是相对来说固定不变的部分,称为静态部分，即后台系统中的模板文件,另外一部分是根据外部输入变化的部分，即来自前端用户的输入（如选择哪些列做数据预处理，选择什么模型构建数据等等），如图2所示。



图2基于模板的自动代码生成

本系统基于数据挖掘的流程，通过预先设置好数据预处理、数据可视化、数据建模对应的模板文件。经过代码生成引擎调用，结合前端数据的输入，动态生成可运行的python脚本，同时运行脚本生成可视化报告，用于前端展示。

为了能够取得更直观的界面以及更佳的用户体验，本文采用Ｄｊａｎｇｏ＋ｍｏｎｇｏＤＢ＋Ｖｕｅ的架构实现代码生成器的系统功能。

用户通过上传数据集，系统读取数据存入mongodb，前端可展示该用户所有数据集，通过预览按钮解析上传文件并做对应前端展示。数据展示端提供一键生成数据分析报告功能，以交互式网页展示所选数据集每一列的类型、缺失值、异常值、强相关变量，并以高亮的形式展现，用以提示用户，作为数据预处理的参考

## 4 系统功能简述

本系统以数据驱动。通过用户上传数据集到用户数据管理，选择需要分析的数据集，进行数据相关操作。

①、数据预览。数据预览将数据集以表格的形式展示，并支持一键生成数据分析报告。数据分析报告针对数据进行整体的探索性分析，将数据集中存在缺失值、异常值、高相关度的关键信息予以高亮展示。同时针对数据集各列的维度进行详细分析以及分布特征的可视化图表。数据分析报告给出的高亮信息用以为数据清洗提供参考方向

②、数据清洗。用户在已上传数据列表选择需要分析的数据集，自定义选择不同的列，提供不同的数据清洗规则。支持的数据清洗规则主要包括缺失值填充、排序、模糊匹配、归一化等，允许用户自定义调换规则顺序，后台引擎根据用户输入的规则进行校验，并展示清洗后的数据表格。

③、数据建模。用户选择指定数据集，选择数据集中的若干列作为数据特征，选择一列作为目标列。提供分类、回归问题的若干种训练模型以及模型评估方法。参数选择完毕通过弹窗显示生成的代码

# 数据挖掘代码生成系统分析

## 1 整体架构分析

### 系统架构图

本系统基于Web框架运行，使用Python采用前后端分离的方式进行。

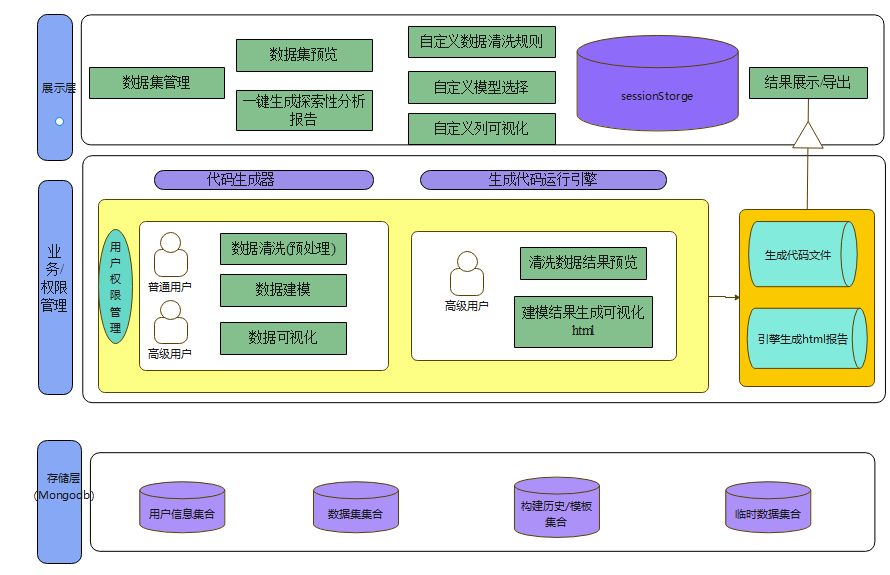


图2-1 数据挖掘代码生成系统架构图

## 2.2 网站需求分析

### 2.2.1 需求分析概述

1.普通用户：普通用户可上传数据集（最大支持上传5份数据集），用户可在上传的数据集中选择需要分析的数据集进行预览，根据预览界面的数据分析报告生成详细数据探索分析报告，该报告为用户进行数据清洗提供指导方向。若数据集无异常数据，则可通过数据建模界面选择指定列需要的模型。根据用户配置，前端弹出窗口展示生成的数据文件。生成的代码支持导出（不支持运行）。

2.高级用户：高级用户可上传数据集（最大支持上传10份数据集），用户可在上传的数据集中选择需要分析的数据集进行预览，根据预览界面的数据分析报告生成详细数据探索分析报告，该报告为用户进行数据清洗提供指导方向。若数据集无异常数据，则可通过数据建模界面选择指定列需要的模型。根据用户配置，前端弹出窗口展示生成的数据文件。生成的代码支持导出，同时

### 网站系统流程图

1. 网站系统流程图：

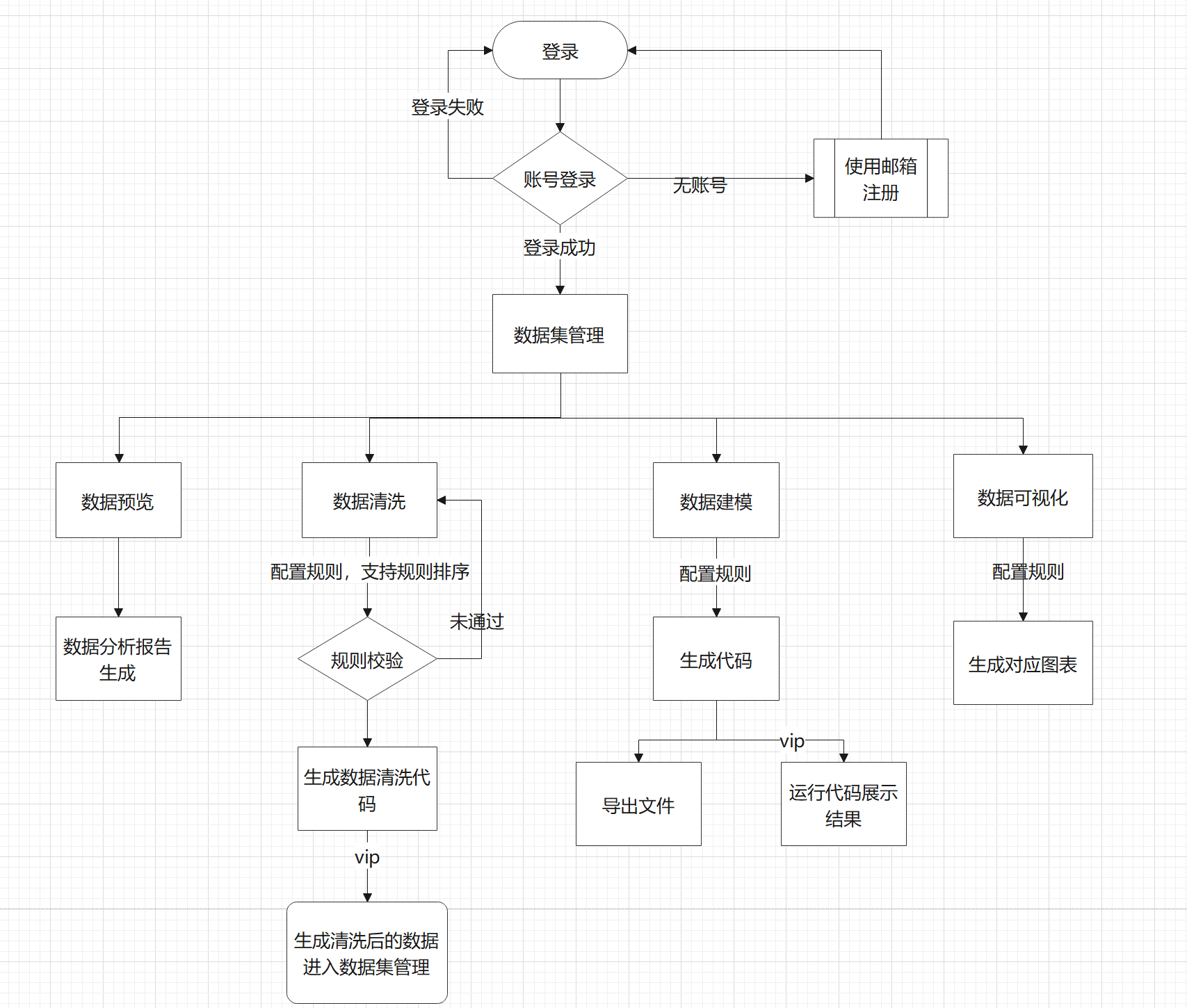


图2-2 代码生成系统流程图

## 2.3 本章小结

本章就网站的整体架构和功能需求进行了详细的介绍和图解。使用internet应用程序的B/S结构来实现网站，采用前后端分离的架构方式，详细的介绍了代码生成系统的存储，引擎，展示结构。并通过流程图的形式来介绍了下整个网站系统的流程，以及各个模块所具有的功能。

# 3 数据库分析与设计-MongoDB

## 3.1 后台数据库以及MongoDB概述

### 3.1.1 Mongodb数据库介绍

MongoDB 是一个基于分布式文件存储的数据库。由 C++ 语言编写。旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。

### 3.1.2 选择MongoDB数据库分析

本系统以数据集为核心构造数据挖掘代码，由于不同的数据集拥有不同的列索引以及不同的数据格式，考虑到平台在代码生成/数据预处理过程中会对频繁访问数据集，同时对数据集的不同列进行不同的操作，如果将数据文件以数据库表的形式存储在关系型数据库，则无法遵循传统关系型数据库的ACID规则。

MongoDB 是一个面向文档存储的数据库。文档存储一般用类似json的格式存储，存储的内容是文档型的。这样也就有机会对某些字段建立索引，实现关系数据库的某些功能，同时拥有更好的可拓展性。

## 3.2 数据集合的具体设计

### 3.2.1 Mongodb数据库设计规范

### 3.2.2 数据库集合设计展示

以下就是本网站系统中所用到的所有数据集合

1. user\_model-用户集合：存储用户信息以及用户上传的数据集列表

表3-1

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 备注 |
| \_id | ObjectId | 默认自增id |
| username | String | 用户名 |
| password | String | 密码(base64加密) |
| email | String | 注册邮箱 |
| isVip | Boolean | 高级用户标识 |
| vipEndtime | Double | Vip到期时间 |
| dataset | Array | 用户上传的数据集名称 |

1. dataset\_model:数据集集合，用来保存整个平台上传的数据集

表3-2

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 备注 |
| \_id | ObjectId | 默认自增id |
| username | String | 用户名 |
| dataset\_name | String | 数据集名称 |
| columns | Array | 数据集的各列名称 |
| data | Object | 数据集内容 |

1. temp:临时集合，用于存储验证码内容

表3-3

|  |  |  |
| --- | --- | --- |
| 字段名称 | 字段类型 | 备注 |
| \_id | ObjectId | 默认自增id |
| address | String | 发送目标地址邮箱 |
| check\_code | Int32 | 验证码 |
| send\_time | Double | 验证码发送时间 |

1. 构建历史集合

表3-4

## 3.3 本章小结

本章就系统的实际应用场景出发进行分析，选择MongoDB进行数据存储，使得数据存储更具备拓展性。针对业务需求，将用户数据集合，数据集集合，临时数据（用于保存用户注册账户的邮件验证码）集合，分别进行存储，以达到降低系统业务之间的耦合度，提升运行效率的目的。

# 4 数据挖掘代码生成器模板代码设计

代码的生成主要采用基于基于模板的代码生成方法。基于模板的代码生成的基本原理是把软件的需求分成两部分,一部分是相对来说固定不变的部分,称为静态部分，即后台系统中的模板文件,另外一部分是根据外部输入变化的部分，即来自前端用户的输入（如选择哪些列做数据预处理，选择什么模型构建数据等等），如图4-1所示。



图4-1基于模板的自动代码生成

一般生成代码需要通过三个维度进行考虑。第一个维度是寻找高可复用性代码，即可以直接使用的代码；第二个维度是有规律的代码（即结构相似的代码），此类型可通过自定义配置进行生成，例如不同的数据集的不同名称，不同字段，字段有着不同的类型，但是读取数据，切割数据，训练模型的代码却是相同的。第三个维度是没有规律的代码，即我们在代码生成中需要考虑的特殊场景。

## 4.1 数据清洗代码生成器

　在数据挖掘领域，经常会遇到的情况是挖掘出来的特征数据存在各种异常情况，如 数据缺失 、数据值异常 等。对于这些情况，如果不加以处理，那么会直接影响到最终挖掘模型建立后的使用效果，甚至是使得最终的模型失效，任务失败。因此，数据清洗在建模之前是必要进行的。

### 4.1.1 常见数据清洗方法简介

数据清洗分为数据的读写、数据的探索与描述、数据简单处理、重复值处理、缺失值处理、异常值处理、文本字符串的处理等。依据上述步骤，本系统提供的代码生成器支持缺失值填充、重复项删除、异常值删除、文本筛选（查询）、数据标准化等操作。

其中缺失值填充的方法包括按0填充、均值填充、众数填充、使用上/下一个数据填充等。数据标准化包括最大-最小规范化，标准差标准化。

关于数据标准化。数据标准化是指将数据按比例缩放，使之落入一个小的特定区间。在某些比较和评价的指标处理中经常会用到，去除数据的单位限制，将其转化为无量纲的纯数值，便于不同单位或量级的指标能够进行比较和加权。

最小-最大规范化也称为离散标准化，是对原始数据的线性变换，将数据值映射到[0, 1]之间。转换公式如下：



最大最小标准化保留了原来数据中存在的关系，是消除量纲和数据取值范围影响的最简单方法。这种处理方法的缺点是若数值集中且某个数值很大，则规范化后各值接近于0，并且将会相差不大。

标准差标准化也称为零-均值规范化，经过处理的数据的均值为0，标准差为1。转化公式为。



其中为原始数据的均值，为原始数据的标准差，是当前用得最多的数据标准化方式。标准差分数可以回答这样一个问题："给定数据距离其均值多少个标准差"的问题，在均值之上的数据会得到一个正的标准化分数，反之会得到一个负的标准化分数。

### 4.1.2 数据清洗代码模板设计

数据清洗的模板相对简单且固定。通过固定的读取方法，对不同数据列调用不同的数据清洗方法，清洗方法通过预定义的参数文件获取并填充。数据清洗的代码均通过pandas库完成。

数据清洗方法我们通过将上一小节提及的清洗方法以json文件形式存储，构造数据清洗方法—方法代码的映射关系，达到快速构建的效果。如下所示。（以下json文件中的df代值通过pandas读取的数据集文件,cols为需要操作的列，其他部分为对应数据清洗方法的调用）

{

"缺失值填充": {

"按0填充": "df[cols]=df[cols].fillna(0)",

"均值填充": "df[cols]=df[cols].fillna(df[cols].mean())",

"众数填充": "df[cols]=df[cols].fillna(df[cols].mode())",

"使用上一个数据填充": "df[cols]=df[cols].fillna(method='pad')",

"使用下一个数据填充": "df[cols]=df[cols].fillna(method='bfill')",

"插值法填充": "df[cols] = df[cols].interpolate()"

},

"重复项删除":"df[cols] = df[cols].dropna()",

"排序": "df.sort\_values(by=cols, inplace= True)",

"筛选": {

"大于": "",

"等于": "",

"小于": "",

"包含": ""

},

"标准化": {

"最大最小规范化": "from sklearn.preprocessing import MinMaxScaler;df[cols]=MinMaxScaler().fit\_transform(df[cols])",

"均值规范化": "from sklearn.preprocessing import StandardScaler;df[cols]=StandardScaler().fit\_transform(df[cols])"

} }

### 4.1.3数据清洗代码生成交互

前端页面通过提供下拉框以供用户选择，通过自定义多条清洗规则（规则形式:对xxx列进行xxx操作，具体包含的操作方法在上节中已做介绍）进行代码生成，具体流程图以及界面原型图如下

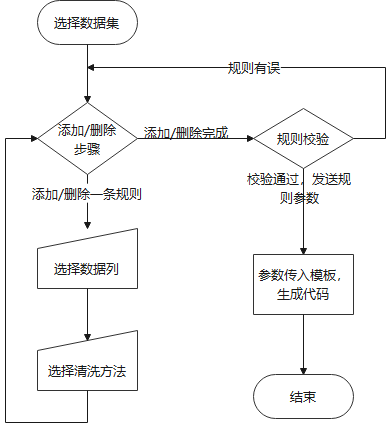


图4-2 数据清洗代码生成交互流程图



图4-3数据清洗代码生成原型图

## 4.2 数据建模（机器学习）代码生成器

数据建模往往离不开机器学习。机器学习的核心思想是创造一种算法，它能从数据中挖掘出有规律的东西，而不需要针对某个问题去写代码。你需要做的只是把数据“投喂”给这个算法，然后它会在数据上建立自己的逻辑。

### 4.2.1 有监督与无监督学习算法

最基本的机器学习算法是解决分类和回归两大类问题,这两类统称为监督学习方法。监督学习通过已有的训练样本去训练得到一个最优模型，再利用这个模型将所有的输入映射为相应的输出，对输出进行简单的判断从而实现预测和分类的目的，也就具有了对未知数据进行预测和分类的能力。就如有标准答案的练习题，然后再去考试，相比没有答案的练习题然后去考试准确率更高。又如我们小的时候不知道牛和鸟是否属于一类，但当我们随着长大各种知识不断输入，我们脑中的模型越来越准确，判断动物也越来越准确。

关于有监督学习方法中分类与回归的区别通过下表进行比对。下表就分类与回归问题的输出类型、目的、评价方法以及应用场景进行对比，如下表 4-2-1所示。下表提及的评价方法的会在下一小节进行说明。

|  |  |  |
| --- | --- | --- |
|  | **分类** | **回归** |
| 输出类型 | 离散数据 | 连续数据 |
| 目的 | 寻找决策边界 | 找到最优拟合 |
| 评价方法 | 准确率，混淆矩阵等 | 均方误差、平均绝对误差、决定系数等 |
| 应用场景 | 分类问题是用于将事物打上一个标签，通常结果为离散值。例如判断一幅图片上的动物是一只猫还是一只狗分类并没有逼近的概念，最终正确结果只有一个。 | 回归问题通常是用来预测一个值，如预测房价、未来的天气情况等等，例如一个产品的实际价格为500元，通过回归分析预测值为499元，我们认为这是一个比较好的回归分析，回归是对真实值的一种逼近预测。 |

表4-2-1 分类与回归模型简介

无监督学习： 对没有概念标签（分类）的训练样本进行学习，以发现训练样本集中的结构性知识。这里，所有的标签（分类）是未知的。因此，训练样本的岐义性高。即我们事先没有任何训练样本，而需要直接对数据进行建模。典型的无监督学习算法是聚类算法。

因此本代码生成器主要是通过构建回归、分类、聚类问题的模板，建立快速生成对应模型需求的代码。

### 4.2.2 模型评估方法简介

在模型建立完成后，我们还需要对模型进行评价来判断建模效果的优劣。因此，我们通过分类、回归、聚类三种模型分别定义了不同的评价指标。

#### 4.2.2.1、分类模型评价指标

在分类模型的评价过程中，我们往往以混淆矩阵为核心进行模型好坏的评估，混淆矩阵如下表所示。

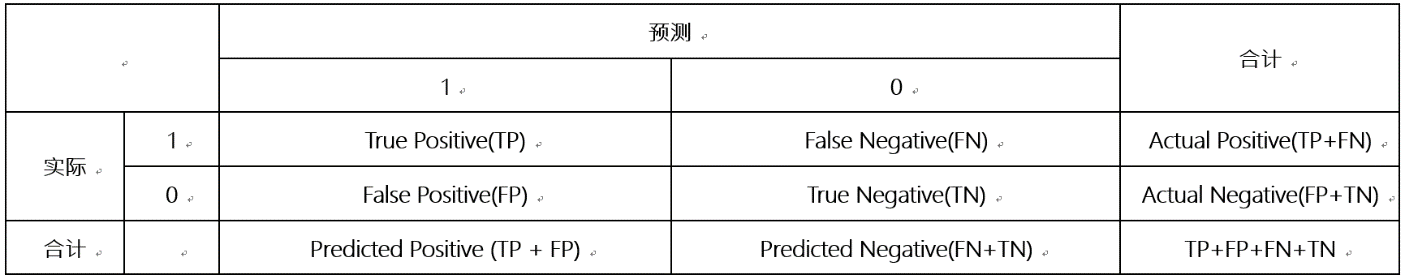


表4-2-2 混淆矩阵

横向为真实结果、纵向为预测结果。对角为预测正确，反对角为预测错误。上表中TP表示将正类预测为正类;FN表示将正类预测为负类FP表示将负类预测为正类TN表示将负类预测为负类。通过这四个值我们可以做出进一步的计算。在本系统中，我们通过混淆矩阵，计算模型的准确率、漏报率、误报率，并依据这些值绘制ROC曲线可视化结果，并计算AUC值对模型的好坏进行评估。

①准确率(Accuracy)：对于给定的测试集，分类模型正确分类的样本数与总样本数之比即Accuracy = (TP+TN)/(TP+TN+FP+FN)。

②漏报率(FPR)：即负正类率，指本来正常（Condition negative），但是误认为是异常（Predicted condition positive）的值，即FPR= FP/(TP+TN+FP+FN)。

③误报率(FNR):本来异常（Condition positive），但是误认为是正常（Predicted condition negative）。即FNR = FN/(TP+TN+FP+FN)。

④ROC曲线

ROC曲线中的主要两个指标就是真正率和假正率，上面也解释了这么选择的好处所在。其中横坐标为假正率(FPR)，纵坐标为真正率(TPR)。一个标准的ROC曲线图如下图4-2-1所示。

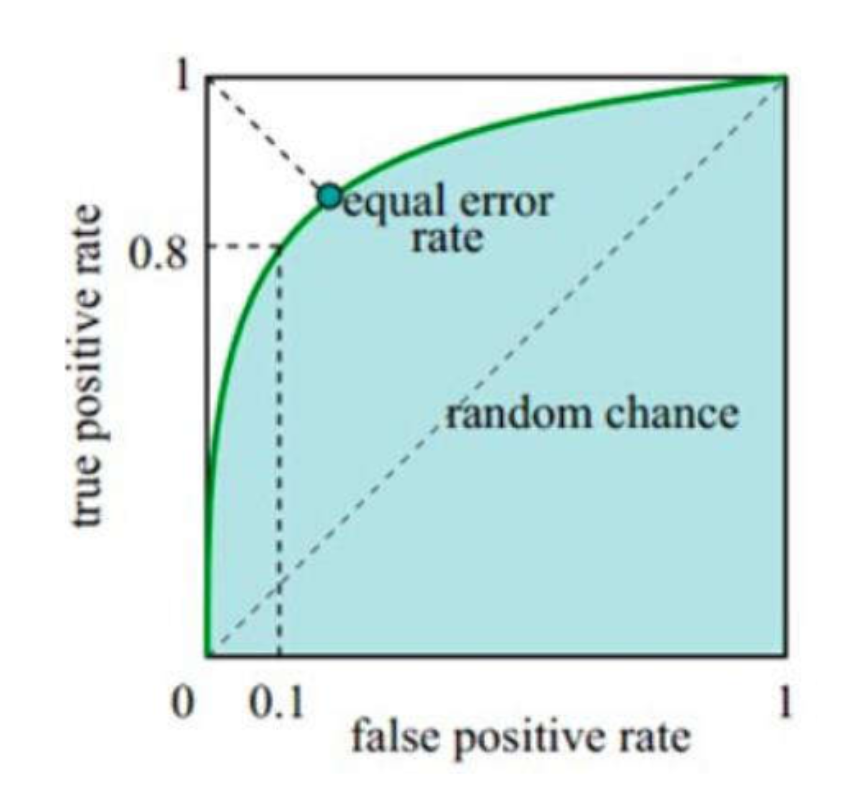


图4-2-1 roc曲线图

根据roc曲线我们可以如此判定一个模型的优劣：FPR表示模型虚报的响应程度，而TPR表示模型预测响应的覆盖程度。因为我们通常期望虚报的越少越好，覆盖的越多越好。即TPR越高，同时FPR越低（即ROC曲线越陡），那么模型的性能就越好。

⑤AUC值

AUC全称**曲线下面积（Area Under Curve）**。这里的曲线也即是上述的ROC曲线。AUC值是一个概率值，当你随机挑选一个正样本以及一个负样本，当前的分类算法根据计算得到的Score值将这个正样本排在负样本前面的概率就是AUC值。通常情况下，AUC数值介于0.5-1之间，可以评价分类器的好坏，数值越大说明越好。

#### 4.2.2.2、回归模型评价指标

（一）平均绝对误差（Mean Absolute Error，MAE）



平均绝对误差就是指预测值与真实值之间平均差值。平均绝对误差能更好地反映预测值误差的实际情况.

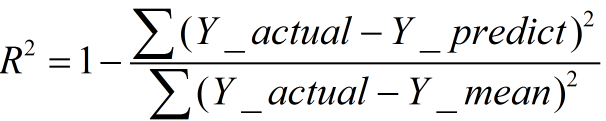
（二）均方误差（Mean Squared Error，MSE）



观测值与真值偏差的平方和与观测次数的比值。这也是线性回归中最常用的损失函数，线性回归过程中尽量让该损失函数最小。那么模型之间的对比也可以用它来比较。

MSE可以评价数据的变化程度，MSE的值越小，说明预测模型描述实验数据具有更好的精确度。

（三）R-square(决定系数)



分母表示原始数据的离散程度，分子表示预测数据和原始数据的误差，二者相除可以消除来自原始数据离散程度的影响。

其实“决定系数”是通过数据的变化来表征一个拟合的好坏。

理论上取值范围（-∞，1], 正常取值范围为[0 1]。实际操作中通常会选择拟合较好的曲线计算R²，因此很少出现-∞。

R²越接近1，表明方程的变量对y的解释能力越强，这个模型对数据拟合的也较好越接近0，表明模型拟合的越差。

缺点：数据集的样本越大，R²越大，因此，不同数据集的模型结果比较会有一定的误差

#### 4.2.2.3、聚类模型评价指标

无监督聚类没有样本输出，也就没有比较直接的聚类评估方法。但是我们可以从簇内的稠密程度和簇间的离散程度来评估聚类的效果。常见的方法有轮廓系数Silhouette Coefficient和Calinski-Harabasz Index。

 1、Calinski-Harabaz Index

Calinski-Harabasz分数值的数学计算公式：

s(k)=

其中m为训练集样本数，k为类别数。Bk为类别之间的协方差矩阵，Wk为类别内部数据的协方差矩阵。tr为矩阵的迹。即类别内部数据的协方差越小越好，类别之间的协方差越大越好，这样的Calinski-Harabasz分数会高

2. 轮廓系数Silhouette coefficient

轮廓系数（Silhouette coefficient）适用于实际类别信息未知的情况。对于单个样本，设a是与它同类别中其他样本的平均距离，b是与它距离最近不同类别中样本的平均距离，轮廓系数为：

s=

对于一个样本集合，它的轮廓系数是所有样本轮廓系数的平均值。轮廓系数取值范围是[−1,1]，同类别样本越距离相近且不同类别样本距离越远，分数越高。

### 4.2.3 数据建模机器学习方法优劣

本文主要围绕数据挖掘的代码生成方法进行展开，对于使用到的各类机器学习的模型原理不做探究，仅针对不同模型的优劣做出对比以供参考。

#### **4.2.3.1､决策树**

优点：

* 1. 决策树易于理解和解释.人们在通过解释后都有能力去理解决策树所表达的意义；
  2. 对于决策树，数据的准备往往是简单或者是不必要的.其他的技术往往要求先把数据一般化，比如去掉多余的或者空白的属性。
  3. 能够同时处理数据型和常规型属性。其他的技术往往要求数据属性的单一。决策树是一个白盒模型。如果给定一个观察的模型，那么根据所产生的决策树很容易推出相应的逻辑表达式。
  4. 易于通过静态测试来对模型进行评测。表示有可能测量该模型的可信度。
  5. 在相对短的时间内能够对大型数据源做出可行且效果良好的结果。
  6. 可以对具有多属性的数据集构造决策树。
  7. 决策树可很好地扩展到大型数据库中，同时它的大小独立于数据库的大小。

缺点

1. 对于各类别样本数量不一致的数据，在决策树当中,信息增益的结果偏向于那些具有更多数值的特征。
2. 决策树处理缺失数据时的困难。
3. 过度拟合问题的出现。
4. 忽略数据集中属性之间的相关性。

#### **4.2.3.2 ANN人工神经网络**

优点：分类的准确度高,并行分布处理能力强,分布存储及学习能力强，对噪声神经有较强的鲁棒性和容错能力，能充分逼近复杂的非线性关系，具备联想记忆的功能等。

缺点：神经网络需要大量的参数，如网络拓扑结构、权值和阈值的初始值；不能观察之间的学习过程，输出结果难以解释，会影响到结果的可信度和可接受程度；学习时间过长,甚至可能达不到学习的目的。

#### **4.2.3.3 KNN最近邻算法**

优点

1. 简单、有效。重新训练的代价较低（类别体系的变化和训练集的变化，在Web环境和电子商务应用中是很常见的）。
2. 计算时间和空间线性于训练集的规模（在一些场合不算太大）。
3. 由于KNN方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于类域的交叉或重叠较多的待分样本集来说，KNN方法较其他方法更为适合。
4. 该算法比较适用于样本容量比较大的类域的自动分类，而那些样本容量较小的类域采用这种算法比较容易产生误分。

缺点

1. KNN算法是懒散学习方法（基本上不学习），因此比一些积极学习的算法要快很多。
2. 类别评分不是规格化的（不像概率评分）。
3. 输出的可解释性不强，例如决策树的可解释性较强。
4. 该算法在分类时有个主要的不足是，当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的K个邻居中大容量类的样本占多数。该算法只计算“最近的”邻居样本，某一类的样本数量很大，那么或者这类样本并不接近目标样本，或者这类样本很靠近目标样本。无论怎样，数量并不能影响运行结果。可以采用权值的方法（和该样本距离小的邻居权值大）来改进。
5. 计算量较大。目前常用的解决方法是事先对已知样本点进行剪辑，事先去除对分类作用不大的样本。

#### 4.2.3.4　SVM支持向量机

优点

* 1. 可以解决小样本情况下的机器学习问题。
  2. 可以提高泛化性能。
  3. 可以解决高维问题。
  4. 可以解决非线性问题。
  5. 可以避免神经网络结构选择和局部极小点问题。

缺点

* 1. 对缺失数据敏感。
  2. 对非线性问题没有通用解决方案，必须谨慎选择Kernelfunction来处理。

#### 4.2.3.5､朴素贝叶斯

优点

* 1. 朴素贝叶斯模型发源于古典数学理论，有着坚实的数学基础，以及稳定的分类效率。
  2. NBC模型所需估计的参数很少，对缺失数据不太敏感，算法也比较简单。

缺点

* 1. 理论上，NBC模型与其他分类方法相比具有最小的误差率。但是实际上并非总是如此，这是因为NBC模型假设属性之间相互独立，这个假设在实际应用中往往是不成立的（可以考虑用聚类算法先将相关性较大的属性聚类），这给NBC模型的正确分类带来了一定影响。在属性个数比较多或者属性之间相关性较大时，NBC模型的分类效率比不上决策树模型。而在属性相关性较小时，NBC模型的性能最为良好。
  2. 需要知道先验概率。
  3. 分类决策存在错误率

#### 4.2.3.6､Adaboost算法

优点

* 1. adaboost是一种有很高精度的分类器。
  2. 可以使用各种方法构建子分类器，Adaboost算法提供的是框架。
  3. 当使用简单分类器时，计算出的结果是可以理解的。而且弱分类器构造极其简单。
  4. 简单，不用做特征筛选。
  5. 不用担心过拟合。

缺点

* 1. AdaBoost迭代次数也就是弱分类器数目不太好设定，可以使用交叉验证来进行确定。
  2. 数据不平衡导致分类精度下降。
  3. 训练比较耗时，每次重新选择当前分类器最好切分点。

#### 4.2.3.7、逻辑回归

优点

* 1. 预测结果是界于0和1之间的概率；
  2. 可以适用于连续性和类别性自变量；
  3. 容易使用和解释；

缺点

* 1. 对模型中自变量多重共线性较为敏感，例如两个高度相关自变量同时放入模型，可能导致较弱的一个自变量回归符号不符合预期，符号被扭转。​需要利用因子分析或者变量聚类分析等手段来选择代表性的自变量，以减少候选变量之间的相关性；
  2. 预测结果呈“S”型，因此从log(odds)向概率转化的过程是非线性的，在两端随着​log(odds)值的变化，概率变化很小，边际值太小，slope太小，而中间概率的变化很大，很敏感。 导致很多区间的变量变化对目标概率的影响没有区分度，无法确定阀值。

#### **4.2.3.8、随机森林**

优点

* 1. 在当前的很多数据集上，相对其他算法有着很大的优势，表现良好
  2. 它能够处理很高维度（feature很多）的数据，并且不用做特征选择
  3. 在训练完后，它能够给出哪些feature比较重要
  4. 在创建随机森林的时候，对generlization error使用的是无偏估计，模型泛化能力强
  5. 训练速度快，容易做成并行化方法
  6. 在训练过程中，能够检测到feature间的互相影响
  7. 实现比较简单
  8. 对于不平衡的数据集来说，它可以平衡误差。
  9. 如果有很大一部分的特征遗失，仍可以维持准确度。

缺点

* 1. 随机森林已经被证明在某些噪音较大的分类或回归问题上会过拟
  2. 对于有不同取值的属性的数据，取值划分较多的属性会对随机森林产生更大的影响，所以随机森林在这种数据上产出的属性权值是不可信的。

#### 4.2.3.9 K-means

k均值聚类算法（k-means clustering algorithm）是一种[迭代](https://baike.baidu.com/item/%E8%BF%AD%E4%BB%A3/8415523)求解的聚类分析算法，其步骤是，预将数据分为K组，则随机选取K个对象作为初始的聚类中心，然后计算每个对象与各个种子聚类中心之间的距离，把每个对象分配给距离它最近的聚类中心。聚类中心以及分配给它们的对象就代表一个[聚类](https://baike.baidu.com/item/%E8%81%9A%E7%B1%BB/593695)。每分配一个样本，聚类的聚类中心会根据聚类中现有的对象被重新计算。这个过程将不断重复直到满足某个终止条件。终止条件可以是没有（或最小数目）对象被重新分配给不同的聚类，没有（或最小数目）聚类中心再发生变化，[误差](https://baike.baidu.com/item/%E8%AF%AF%E5%B7%AE/738024)[平方和](https://baike.baidu.com/item/%E5%B9%B3%E6%96%B9%E5%92%8C/783894)局部最小。

优点:

a) 原理比较简单，实现也是很容易，收敛速度快。

b) 聚类效果较优。

* 1. 算法的可解释度比较强。
  2. 主要需要调参的参数仅仅是簇数k。

缺点:

1. K值的选取不好把握
2. 对于不是凸的数据集比较难收敛
3. 如果各隐含类别的数据不平衡，比如各隐含类别的数据量严重失衡，或者各隐含类别的方差不同，则聚类效果不佳。
4. 采用迭代方法，得到的结果只是局部最优。
5. 对噪音和异常点比较的敏感。

#### 4.2.3.10 BRICH聚类算法

BIRCH算法可以不用输入类别数K值，这点和K-Means，Mini Batch K-Means不同。一般来说，BIRCH算法适用于样本量较大的情况，这点和Mini Batch K-Means类似，但是BIRCH适用于类别数比较大的情况，而Mini Batch K-Means一般用于类别数适中或者较少的时候。BIRCH除了聚类还可以额外做一些异常点检测和数据初步按类别规约的预处理。但是如果数据特征的维度非常大，比如大于20，则BIRCH不太适合，此时Mini Batch K-Means的表现较好

BIRCH算法的主要优点有：

1. 节约内存，所有的样本都在磁盘上。
2. 聚类速度快，只需要一遍扫描训练集就可以建立CF Tree。
3. 可以识别噪音点，还可以对数据集进行初步分类的预处理

BIRCH算法的主要缺点有：

1. 聚类的结果可能和真实的类别分布不同.
2. 对高维特征的数据聚类效果不好。此时可以选择Mini Batch K-Means
3. 如果数据集的分布簇不是类似于超球体，或者说不是凸的，则聚类效果不好。

#### 4.2.3.11 DBSCAN 算法

DBSCAN 算法是一种基于密度的聚类算法：1.聚类的时候不需要预先指定簇的个数2.最终的簇的个数不确定。DBSCAN算法将数据点分为三类： 1.核心点：在半径Eps内含有超过MinPts数目的点;2.边界点：在半径Eps内点的数量小于MinPts,但是落在核心点的邻域内的点;3.噪音点：既不是核心点也不是边界点的点。

和传统的K-Means算法相比，DBSCAN最大的不同就是不需要输入类别数k，当然它最大的优势是可以发现任意形状的聚类簇，而不是像K-Means，一般仅仅使用于凸的样本集聚类。同时它在聚类的同时还可以找出异常点，这点和BIRCH算法类似。一般来说，如果数据集是稠密的，并且数据集不是凸的，那么用DBSCAN会比K-Means聚类效果好很多。如果数据集不是稠密的，则不推荐用DBSCAN来聚类。

DBSCAN的主要优点有：

* 1. 可以对任意形状的稠密数据集进行聚类，相对的，K-Means之类的聚类算法一般只适用于凸数据集。
  2. 可以在聚类的同时发现异常点，对数据集中的异常点不敏感。
  3. 聚类结果没有偏倚，相对的，K-Means之类的聚类算法初始值对聚类结果有很大影响。

DBSCAN的主要缺点有：

* 1. 如果样本集的密度不均匀、聚类间距差相差很大时，聚类质量较差，这时用DBSCAN聚类一般不适合。
  2. 如果样本集较大时，聚类收敛时间较长，此时可以对搜索最近邻时建立的KD树或者球树进行规模限制来改进。
  3. 调参相对于传统的K-Means之类的聚类算法稍复杂，主要需要对距离阈值ϵ，邻域样本数阈值MinPts联合调参，不同的参数组合对最后的聚类效果有较大影响。

### 4.2.4 数据建模（机器学习）代码生成模板设计

在生成数据建模代码之前，我们需要给用户提供代码生成的指导方向，即需要传入的参数。数据建模时我们往往需要知道数据集的特征列、目标列以构建合适的模型解决问题。因此特征列，目标列以及模型类型是代码生成器的必要参数，基于此，我们的代码生成器构建了如下的操作流程：

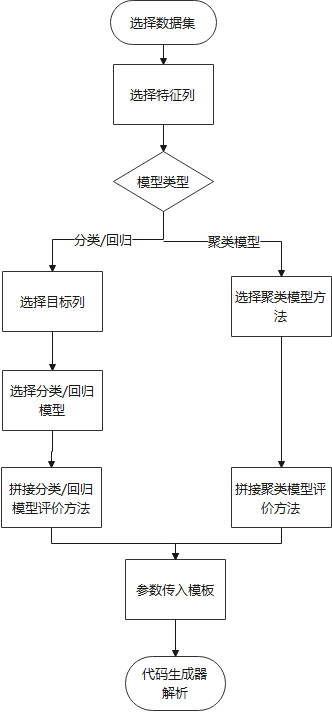


图4-2-4-1 数据建模代码生成器用户流程图

用户传入参数后至代码生成器后台，后台将参数解析，并在对应模板中扩充代码。传递的参数主要包括数据集名称，模型类型（分类、回归或者聚类），特征列名称，目标列名称（仅分类，回归任务包含），应用模型（一个或者多个）等参数传至后台数据建模的代码生成的逻辑主要如下图所示。

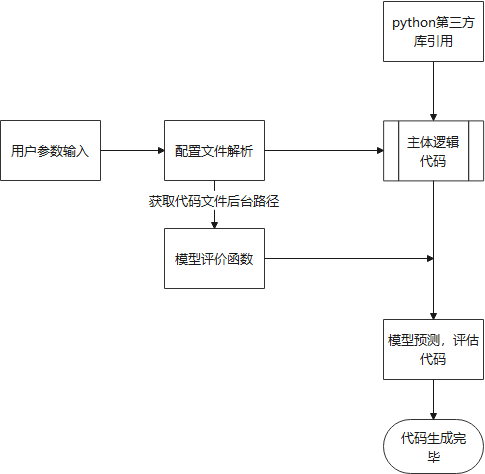


图4-2-4 数据建模代码生成逻辑流程图

首先需要引入建模需要的python第三方库，包括用于数据分析的pandas，numpy，用于机器学习建模的sklearn。

通过读取用户的参数输入获取需要构建的模型类型，进行主体逻辑的代码拼接。主体代码逻辑包括读取数据集文件、选择特征列，目标列，并拆分数据集为训练集与测试集。拆分完成后，将训练集作为模型的输入进行训练，并将训练的好的模型（模型支持多选）以列表的形式保存，作为后续预测模型方法的输入。

根据用户选择的分类/回归/聚类任务下的不同模型，进行主体逻辑拼接。生成器获取模型的映射关系以json形式储存，如下key-value结构所示。其中key为具体模型的名称，value为该python调用sklearn中封装的api的具体代码，在生成代码的过程中会优先导入这部分方法，导入方法完成后会对这部分字符串进行分隔，转换成python中实际运行的语法：

{

'分类': {

'朴素贝叶斯': 'from sklearn.naive\_bayes import GaussianNB',

'决策树': 'from sklearn.tree import DecisionTreeClassifier',

'支持向量机': 'from sklearn.svm import SVC',

'神经网络': 'from sklearn.neural\_network import MLPClassifier',

'逻辑回归': 'from sklearn.linear\_model import LogisticRegression',

'KNN': 'from sklearn.neighbors import KNeighborsClassifier',

"Adaboost": 'from sklearn.ensemble import AdaBoostClassifier',

"随机森林": 'from sklearn.ensemble import RandomForestClassifier'

},

'回归': {

'线性回归': 'from sklearn.linear\_model import LinearRegression',

'KNN': 'from sklearn.neighbors import KNeighborsRegressor',

'决策树': 'from sklearn.tree import DecisionTreeRegressor',

'支持向量机': 'from sklearn.svm import SVR',

'神经网络': 'from sklearn.neural\_network import MLPRegressor',

'Adaboost': 'from sklearn.ensemble import AdaBoostRegressor',

'随机森林':'from sklearn.ensemble import RandomForestRegressor'

},

'聚类': {

'K\_means': 'from sklearn.cluster import KMeans',

'MiniBatchKMeans': 'from sklearn.cluster import MiniBatchKMeans',

'BIRCH': 'from sklearn.cluster import Birch'

},

此外，针对分类模型与回归模型分别有着不同的模型评价参数，计算过程代码以单独模块的形式保存在后台。通过任务类型读取模块文件，将文件内容拼接于主体逻辑代码之后。至此，数据建模的代码生成已经完成。

## 4.3 本章小结

本章采用基于基于模板的代码生成方法，通过研究数据清洗、数据建模的流程就行关键参数的提取，用以动态嵌入模板生成代码。首先就常见的数据清洗方法，计算过程，以及数据清洗的模板设计思路做了介绍。接着在已有无异常缺失值的“净”数据的前提下，就常用的回归、建模和聚类的机器学习方法以及分别对应的评价指标做了介绍，并就各种模型方法的优点与缺点做出了详尽的对比。基于模型的理解以及评价指标的不同，将构建模型必要的参数（模型类型，特征列名称，目标列名称，应用模型）进行提取，动态插入到设计好的主体逻辑代码中生成完整代码。

# 5 数据分析系统引擎设计

## 5.1 数据清洗引擎设计

由于不同的数据集有着不同的清洗步骤和方法，这意味着针对不同来源的数据所遇到的困难也是不同的。因此，固定顺序的数据清洗模板往往很难应用于所有的数据集，数据清洗引擎的功能主要是使得用户能够自定义进行数据清洗规则的排序，同时根据数据清洗规则进行处理，对规则的合理性进行校验，对于能生效的清洗规则，输出对应的新数据，而不仅仅是生成一份看似合理的清洗代码。

所以，数据清洗引擎主要在于规则合理性的校验，当所有清洗规则校验通过时，生成一份新的数据集，即清洗完毕的数据集。对于校验不通过的规则，则解析后台错误信息，在前端给出具体错误原因。基于此，设计如下引擎的工作流程如下。

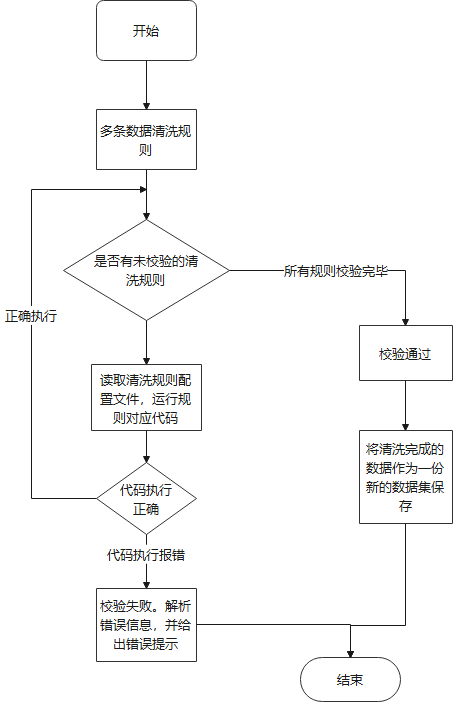


图5-1 数据清洗引擎流程图

## 5.2 数据建模（机器学习）引擎设计

在用户上传不同的数据集文件时，引擎通过读取数据集内容判断需要使用的模型。这里我们定义如下规则:首先看数据的样本是否大于 50。如果样本小于 50 则需要收集更多的样本数据。如果样本大于 50 就先判断预测的是否是一个分类，如果预测的是一个分类，就判断预测的数据是否有标签，如果预测的数据有标签就选择分类模型，否则就选择聚类模型。

当代码生成器运行生成了代码，引擎主要负责运行代码，并根据运行结果生成一份报告，用以展示模型结果。当有多个模型存在时，展示各个模型的结果对比。

由于生成的代码从使用者的角度出发，因此读取数据的步骤是通过加载本地数据文件到内存进行后续建模操作。考虑到数据建模引擎运行在后端，可通过数据库读取数据集文件，为了节省服务器的资源开销，引擎运行文件前会修改原始生成代码的数据源，即从本地文件系统加载修改为数据库读取。建模引擎的工作原理如下

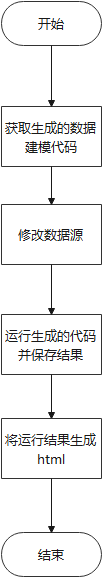


图5-2数据建模引擎流程图

## 5.3 本章小结

在代码生成器已经生成代码的基础上。本章分别针对数据清洗代码，数据建模代码设计运行引擎。数据清洗运行引擎主要通过对用户自定义的清洗规则进行逐条校验，将校验失败的规则展示给用户，并给出对应修改建议。数据建模运行引擎通过执行生成代码，并依据模型类型生成模型评估图表，当存在多个模型时，给出多个模型的效率对比以及关键评价参数的对比可视化图表，生成一份评估报告。使得代码生成器的系统拥有更好的交互，以提升易用性。此外，考虑到运行引擎会占用较多的服务器性能，因此在引擎的使用层面做了相对应 的权限控制，仅高级用户可以使用。

# 参考文献

# 致 谢

从17年的秋天进入大学，学会输出第一行“Hello,World”起，便开始了我对计算机世界的探索。经过这四年的学习、研究、探索，对课本教材知识点，以及时下技术热点的理解，想通过毕业设计这样一个途径，以独立完成这样一个完整的代码生成系统的方式，来表达老师们这几年来的悉心栽培的感激之情。短短四年时间，是老师们实验课上一次次断点调试的帮助，是课堂上上至对编程架构的讲解，对前沿科技的深刻见解，下至内存中每个字节的存放，机器语言对1和0 的每一次编排让我被软件工程，数据科学这些技术的优雅的实现方式所折服。此外，也是每一次竞赛，每一次跟着老师做项目的经历，让我对那些教材上看似简短，实则凝练至极的概念有了深层次的领悟。

其次，要感谢我的父母，对我求学路上每一步的鼓励与肯定，是我最坚强的后盾。感谢一起同窗四年的室友，是课堂上共同进步，课下一起研究技术，在数不清多少个熄灯的夜晚，一起攻克学习新知识的瓶颈，完成了一次次自我的突破。是假期一起旅游，游览大江大河，回到学校想通过更加努力的学习去见识更加广阔的世界。

最后我要感谢母校的栽培，给我提供宝贵的学习资源，以及这样一个广阔的平台感谢数计学院。这些经历都是我人生中宝贵的财富，是弥足珍贵的纪念品，好好收起，仍热满怀热情，继续前行。