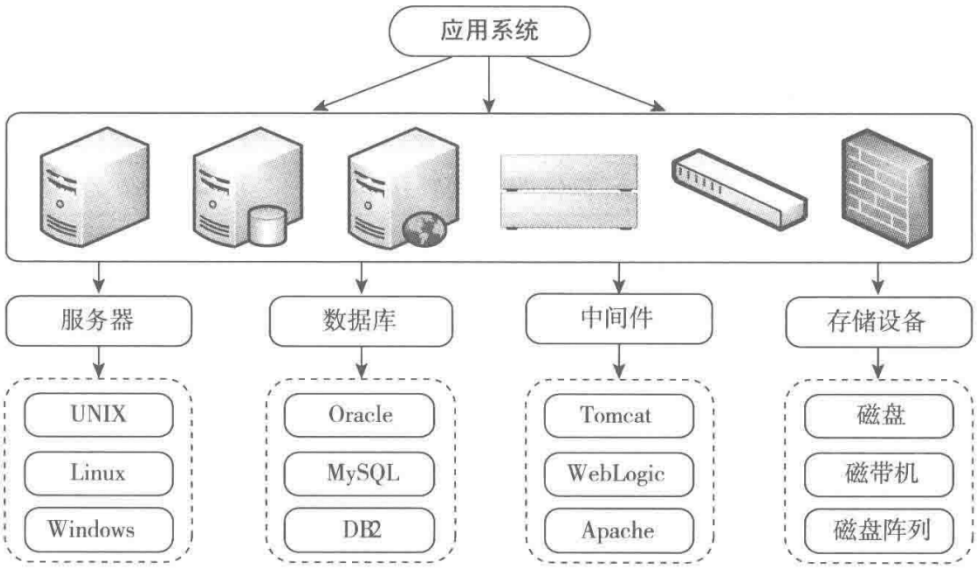
应用系统负载分析与磁盘容量预测

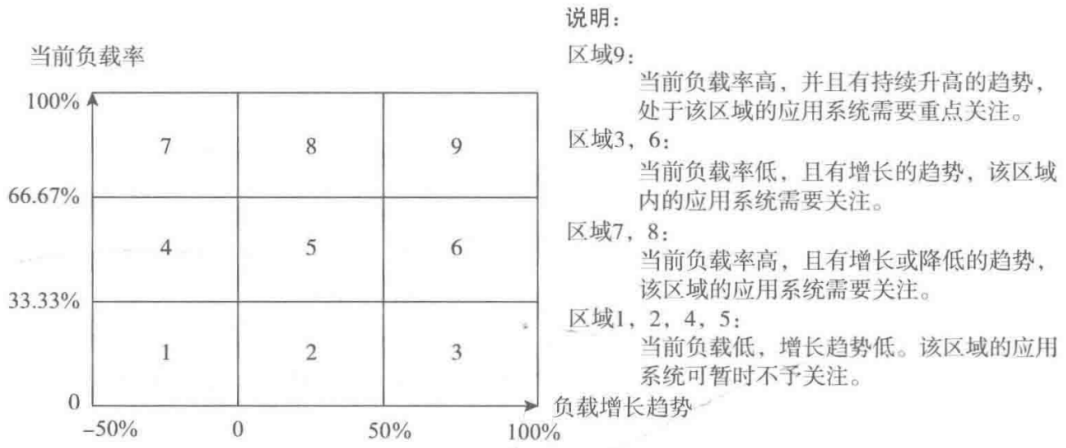
项目背景

- 某大型企业为了信息化发展的需要，建设了办公自动化系统，人力资源管理系统，财务管理系统，企业信息门户系统等几大企业级应用系统。因应用系统在日常运行时，会对底层软硬件造成负荷，显着影响应用系统性能。
- 一般认为，影响应用系统性能的因素包括：服务器，数据库，中间件和存储设备任何一种资源负载过大，都可能会引起应用系统性能下降甚至瘫痪因此，需要关注服务器，数据库，中间件和存储设备的运行状态，及时了解当前应用系统的负载情况，以便提前预防，确保系统安全稳定运行。



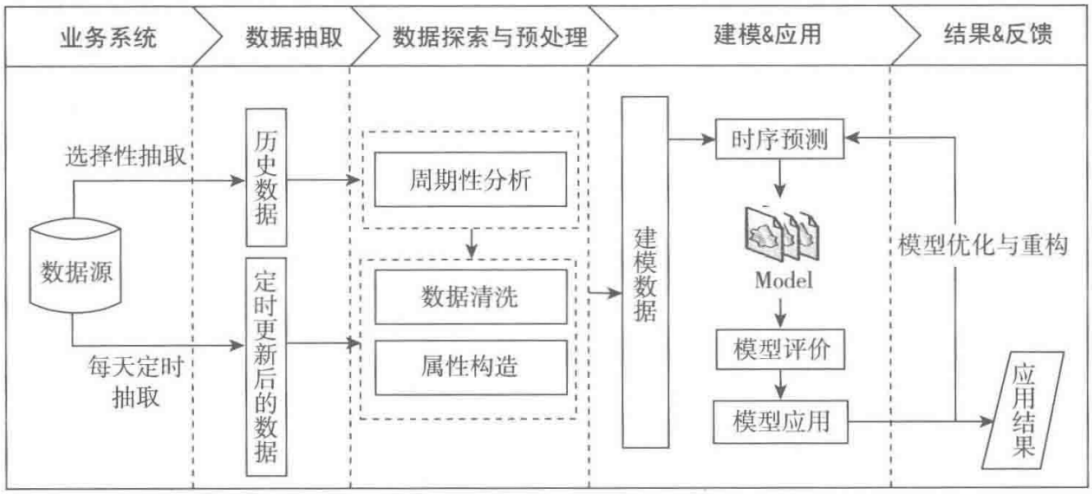
项目需求

- 应用系统的负载率可以通过对一段时间内软硬件性能的运行情况进行综合评分而获得。通过对系统的当前负载率与历史平均负载率进行比较，获得负载率的当前趋势。通过负载率以及负载趋势可对系统进行负载分析。当出现应用系统的负载高或者负载趋势大的情况，代表系统目前处于高危工作环境中。如果系统管理员不及时进行相应的处理，系统很容易出现故障，从而导致用户无法访问系统，严重影响企业的利益。
- 本案例重点分析存储设备中磁盘容量预测，通过对磁盘容量进行预测，可以预测磁盘未来的负载情况，避免应用系统因出现存储容量耗尽的情况而导致应用系统负载率过高，最终引发系统故障。
- 目前监控采集的性能数据主要包含CPU使用信息，内存使用信息和磁盘使用信息等。通过分析磁盘容量相关数据，预测应用系统服务器磁盘空间是否满足系统健康运行的要求。实现以下目标。
    - 针对历史磁盘数据，采用时间序列分析方法，预测应用系统服务器磁盘已使用空间大小。
    - 根据用户需求设置不同的预警等级，将预测值与容量值进行比较，对其结果进行预警判断，为系统管理员提供定制化的预警提示。



项目简介

本案例可采用时间序列分析法对磁盘已使用空间进行预测分析。



- 从数据源中选择性抽取历史数据与每天定时抽取数据。
- 对抽取的数据进行周期性分析以及数据清洗，数据变换等操作后，形成建模数据。
- 采用时间序列分析法对建模数据进行模型构建，利用模型预测服务器磁盘已使用情况。
- 应用模型预测服务器磁盘将要使用情况，通过预测到的磁盘使用大小与磁盘容量大小按照定制化标准进行判断，将结果反馈给系统管理员，提示管理员需要注意磁盘的使用情况。

项目实现步骤

数据获取

- 磁盘使用情况的数据都存放在性能数据中，而监控采集的性能数据中存在大量的其他属性数据。为了抽取出磁盘数据，以属性的标识号（TARGET_ID）与采集指标的时间（`COLLECTTIME`）为条件，对性能数据进行抽取。本案例抽取2014-10-01到情节中字财务管理系统中某一台数据库服务器的磁盘相关数据。

属性名称	属性说明	属性名称	属性说明
SYS_NAME	资产所在的系统名称	ENTITY	具体的属性
NAME	资产名称	VALUE	采集到的值
TARGET_ID	属性的标识号 183 表示磁盘容量大小 184 表示磁盘已使用大小	COLLECTTIME	采集的时间
DESCRIPTION	针对属性标识的说明		

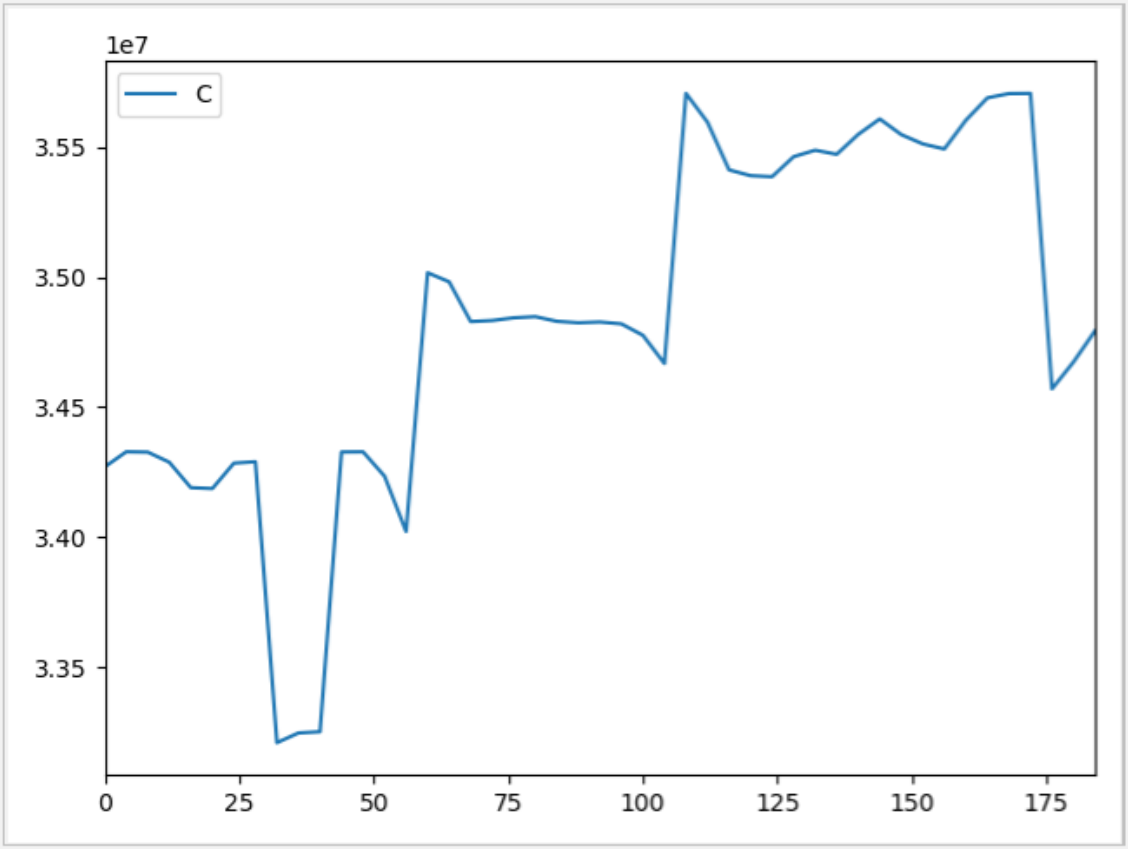
数据探索

使用时序分析法进行建模，为了建模需要，需要探索数据的平稳性。通过时序图可以初步发现数据的平稳性。针对磁盘已使用大小，以天为单位进行周期性分析。

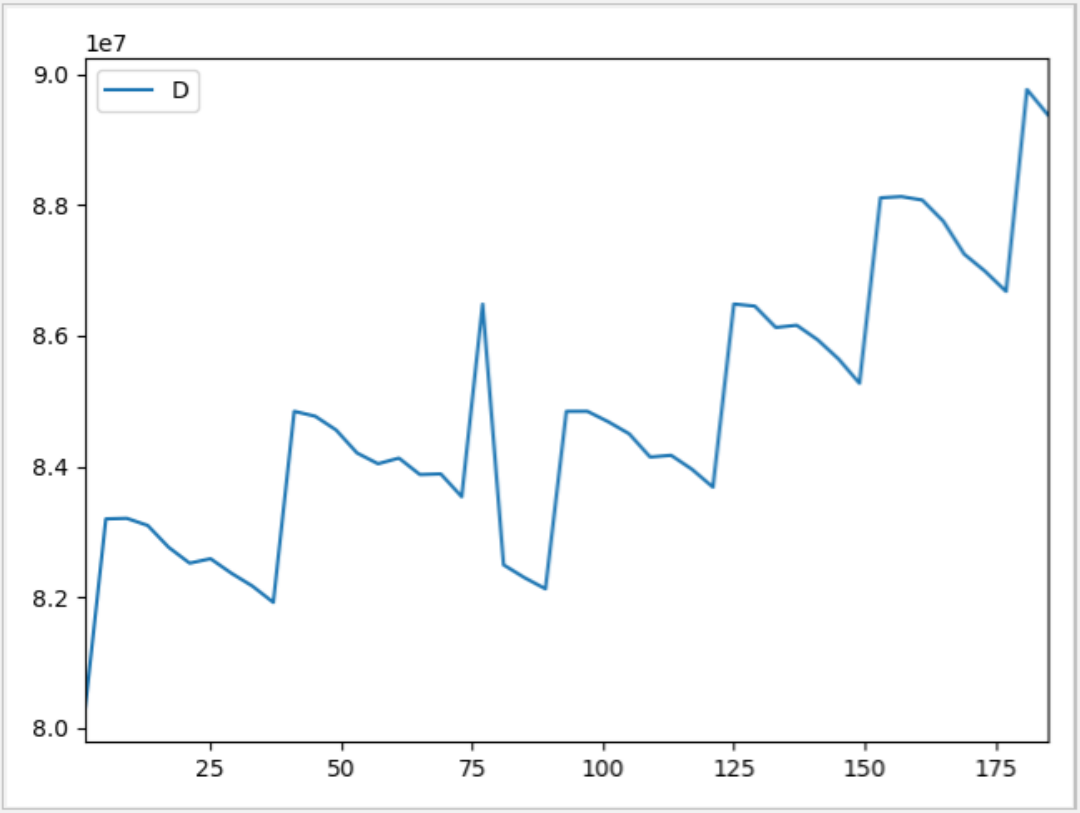
```
# -*- coding:utf-8 -*-
import pandas as pd
import matplotlib.pyplot as plt
def draw():
    data = pd.read_excel('data/discdata.xls')
    str1 = 'C:\\'
    str2 = 'D:\\'
    dataC = data[(data['DESCRIPTION'] == '磁盘已使用大小') & (data['ENTITY'] ==
str1)]
    dataD = data[(data['DESCRIPTION'] == '磁盘已使用大小') & (data['ENTITY'] ==
str2)]
    dataC.plot(y='VALUE', label='C')
    dataD.plot(y='VALUE', label='D')
    plt.show()

if __name__ == '__main__':
    draw()
```

运行结果如下:

C盘磁盘占用率展示:



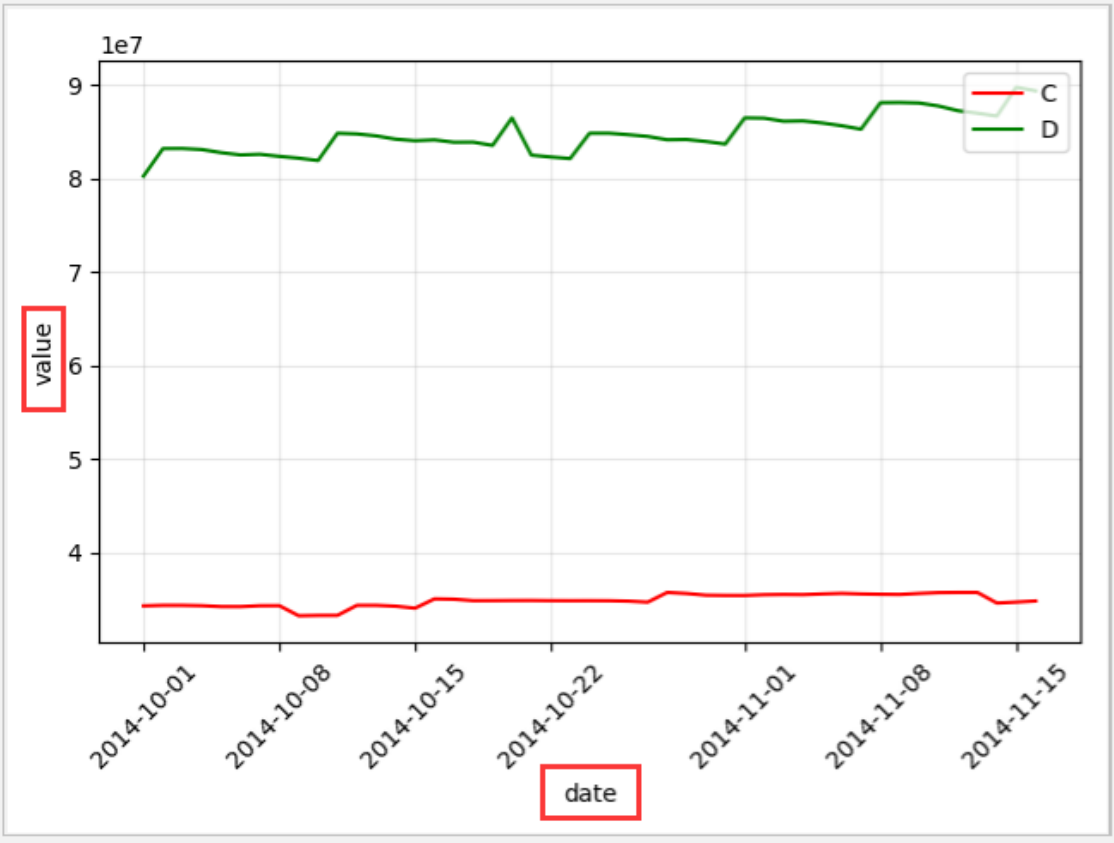
D盘磁盘占用率展示:



更加友好的数据探索图:

```
def draw1():
    from pandas.plotting import register_matplotlib_converters
    register_matplotlib_converters()
    data = pd.read_excel('data/discdata.xls')
    str1 = 'C:\\'
    str2 = 'D:\\'
    dataC = data[(data['DESCRIPTION'] == '磁盘已使用大小') & (data['ENTITY'] ==
str1)]
    dataD = data[(data['DESCRIPTION'] == '磁盘已使用大小') & (data['ENTITY'] ==
str2)]
    plt.plot(dataC['COLLECTTIME'], dataC['VALUE'], label='C', color='r')
    plt.plot(dataD['COLLECTTIME'], dataD['VALUE'], label='D', color='g')
    plt.xlabel('date')
    plt.ylabel('value')
    plt.xticks(rotation=45)
    plt.grid(alpha=0.3)
    plt.legend(loc='upper right')
    plt.show()


if __name__ == '__main__':
    draw1()
```

运行结果如下:



数据预处理

- 数据清洗

  实际的业务中，监控系统会每天定时对磁盘的信息进行收集，但是一般情况下，磁盘的容量属性都是一个定值，因此磁盘原始数据中会存在磁盘容量的重复数据。在数据清洗的过程中，剔除磁盘容量的重复数据，并且将所有服务器的磁盘容量作为一个固定值，方便模型预警。

- 属性构造

  由于每台服务器的磁盘信息可以通过NAME，TARGET_ID，ENTITY三个属性进行区分，且每台服务器的上述三个属性是不变的，可以进行合并。

源数据:

SYS_NAME	NAME	TARGET	DESCRIPTION	ENTITY	VALUE	COLLECTTIME
财务管理系统	CWXT_DB	184	磁盘已使用大小	C:\	34270787.33	2014/10/1
财务管理系统	CWXT_DB	184	磁盘已使用大小	D:\	80262592.65	2014/10/1
财务管理系统	CWXT_DB	183	磁盘容量	C:\	52323324	2014/10/1
财务管理系统	CWXT_DB	183	磁盘容量	D:\	157283328	2014/10/1
财务管理系统	CWXT_DB	184	磁盘已使用大小	C:\	34328899.02	2014/10/2
财务管理系统	CWXT_DB	184	磁盘已使用大小	D:\	83200151.65	2014/10/2
财务管理系统	CWXT_DB	183	磁盘容量	C:\	52323324	2014/10/2
财务管理系统	CWXT_DB	183	磁盘容量	D:\	157283328	2014/10/2
财务管理系统	CWXT_DB	184	磁盘已使用大小	C:\	34327553.5	2014/10/3
财务管理系统	CWXT_DB	184	磁盘已使用大小	D:\	83208320	2014/10/3
财务管理系统	CWXT_DB	183	磁盘容量	C:\	52323324	2014/10/3

合并后的数据:

```
SYS_NAME,CWXT_DB:184:C:\,CWXT_DB:184:D:\ COLLECTTIME
财务管理系统,34270787.33,80262592.65,2014-10-01
财务管理系统,34328899.02,83200151.65,2014-10-02
财务管理系统,34327553.5,83208320.0,2014-10-03
财务管理系统,34288672.21,83099271.65,2014-10-04
财务管理系统,34190978.41,82765171.65,2014-10-05
财务管理系统,34187614.43,82522895.0,2014-10-06
财务管理系统,34285280.22,82590885.0,2014-10-07
财务管理系统,34290578.41,82368173.3,2014-10-08
财务管理系统,33211870.4,82172263.3,2014-10-09
财务管理系统,33249253.87,81922685.0,2014-10-10
财务管理系统,33253832.53,84844722.95,2014-10-11
```

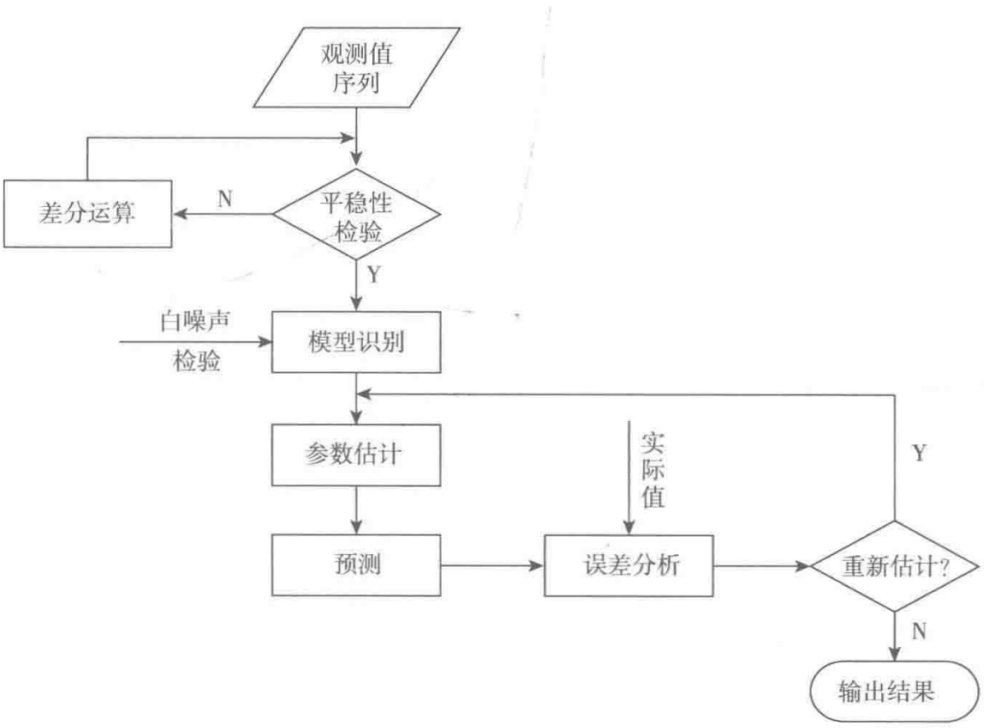
代码实现如下:

```
# -*-coding: utf-8-*-
import xlwt
import pandas as pd
def attr_trans(x):
```

```
    """
    x的内容如下:
           SYS_NAME     NAME       TARGET_ID    DESCRIPTION         ENTITY
VALUE          COLLECTTIME
    184    财务管理系统   CWXT_DB    184          磁盘已使用大小      C:\
34793245.31     2014-11-16
    185    财务管理系统   CWXT_DB    184          磁盘已使用大小      D:\
89377527.25     2014-11-16
    """
    result = pd.Series(index=['SYS_NAME', 'CWXT_DB:184:C:\\',
'CWXT_DB:184:D:\\', 'COLLECTTIME'])
    result['SYS_NAME'] = x['SYS_NAME'].iloc[0]
    result['COLLECTTIME'] = x['COLLECTTIME'].iloc[0]
    result['CWXT_DB:184:C:\\'] = x['VALUE'].iloc[0]
    result['CWXT_DB:184:D:\\'] = x['VALUE'].iloc[1]
    return result

if __name__ == '__main__':
    discfile = 'data/discdata.xls'
    transformeddata = 'data/discdata_processed.csv'
    # 加载excel文件
    data = pd.read_excel(discfile)
    # 目前只对target_id=184的数据进行处理;
    data = data[data['TARGET_ID'] == 184].copy()
    # 按时间分组
    data_group = data.groupby('COLLECTTIME')
    #  # 将函数应用到由各列或行形成的数组上。DataFrame的apply方法可以实现此功能
    data_processed = data_group.apply(attr_trans)
    data_processed.to_csv(transformeddata, index=False)
```

数据挖掘建模

建模流程图:



- 数据集划分

  最后5条记录作为验证数据，其余作为建模样本数据。

- 平稳性检验

  为了确定原始数据序列中没有随机趋势或确定趋势，需要对数据进行平稳性检验，否则将会产生“伪回归”现象。本案例采用单位根检验（`ADF`）的方法进行平稳性检验。

  使用现成的工具`statsmodels`来实现`ADF`检验

- 白噪声检验

  为了验证序列中有用的信息是否已被提取完毕，需要对噪声进行白噪声检验。如果序列检验为白噪声序列，就说明序列中的有用信息已被提取完毕，剩下的全是随机扰动，无法进行预测和使用。本案例使用LB统计量的方法进行白噪声检验。

- 模型识别

  采用极大似然比方法进行模型的参数估计，估计各个参数的值。然后针对各个不同模型，采用BIC信息准则对模型进行定阶，确定P，Q参数，从而选择最优模型。

- 模型检验

  模型确定后，检验其残差序列是否为白噪声。如果不是白噪声，说明残差中还存在有用的信息，需要修改模型或者进一步提取。本案例所确定的`ARIMA`（0,1,1）模型成功通过检验。

- 模型预测

  应用通过检测的模型进行预测，获取未来5天的预测值，并且与实际值作比较，也就是建模忽略的最后5个数据。

- 模型评价

  为了评价时序预测模型效果的好坏，本案例采用3个衡量模型预测精度的统计量指标：平均绝对误差，均方根误差和平均绝对百分误差结合业务分析，误差阈值设定为1.5。

模型应用

- 每天抽取磁盘数据。
- 对抽取数据进行清洗等预处理。
- 将预处理后的数据存放到模型的初始数据中，得到模型的输入数据，调用模型后5天预测。
- 预测值与实际值对比。
- 调整阈值设置预警。