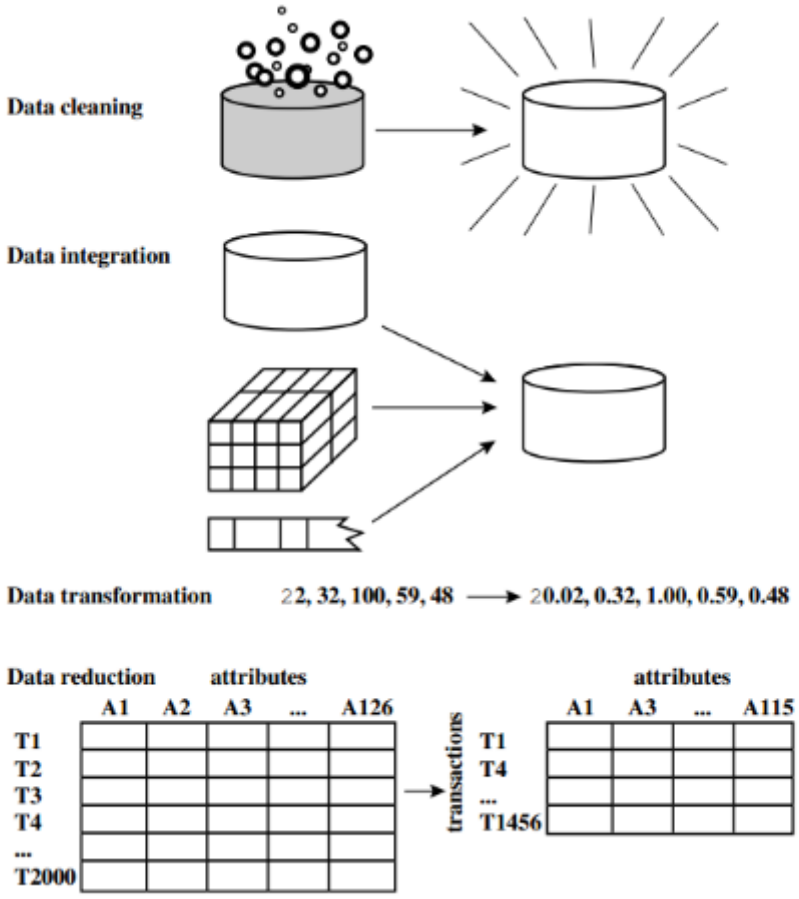
数据采集

利用各种手段获取数据，数据样式不限制，但一般而言是形如`excel`或者`csv`这样的表格格式。

- 数据采集: `urllib`, `requests`
- 数据解析: `Xpath`, `BS4`, 正则表达式
- 数据持久化存储: `pd.to_csv`, `pd.to_excel`, `MySQL`, `Redis`

数据预处理

- 一般而言，数据分析和数据挖掘领域的处理的数据都是海量的数据，这样的数据难免会出现问题。
- 数据预处理占到数据挖掘工作的60%，这是最重要也是最核心的领域.
- 数据预处理分为数据清洗、数据集成、数据变换和数据规约。



数据清洗

数据清洗就是删除无关数据、重复数据、平滑噪声数据、处理缺失值和异常值。

数据缺失的原因

- **有些信息暂时无法获取。**例如在医疗数据库中，并非所有病人的所有临床检验结果都能在给定的时间内得到，就致使一部分属性值空缺出来。又如在申请表数据中，对某些问题的反映依赖于对其他问题的回答。
- **有些信息是被遗漏的。**可能是因为输入时认为不重要、忘记填写了或对数据理解错误而遗漏，也可能是由于数据采集设备的故障、存储介质的故障、传输媒体的故障、一些人为因素等原因而丢失了。

- **有些对象的某个或某些属性是不可用的。**也就是说，对于这个对象来说，该属性值是不存在的，如一个未婚者的配偶姓名、一个儿童的固定收入状况等。
- **有些信息（被认为）是不重要的。**如一个属性的取值与给定语境是无关的，或训练数据库的设计者并不在乎某个属性的取值（称为`dont-care value`）。
- **获取这些信息的代价太大。**
- **系统实时性能要求较高**，即要求得到这些信息前迅速做出判断或决策。

数据缺失的类型

数据集中不含缺失值的变量称为**完全变量**，数据集中含有缺失值的变量称为**不完全变量。**

从缺失的分布来将缺失可以分为**完全随机缺失**，**随机缺失**和**完全非随机缺失。**

- **完全随机缺失（missing completely at random,`MCAR`）：**指的是数据的缺失是完全随机的，不依赖于任何不完全变量或完全变量，不影响样本的无偏性，如家庭地址缺失；
- **随机缺失(missing at random,MAR)：**指的是数据的缺失不是完全随机的，即该类数据的缺失依赖于其他完全变量，如财务数据缺失情况与企业的大小有关；
- **非随机缺失(missing not at random,`MNAR`)：**指的是数据的缺失与不完全变量自身的取值有关，如高收入人群不愿意提供家庭收入；

随机缺失和非随机缺失，直接删除记录是不合适的，随机缺失可以通过已知变量对缺失值进行估计，而非随机缺失的非随机性还没有很好的解决办法。

清洗案例

我给你一组数据，如果要你做数据清洗，你会怎么做？

姓名	语文	英语	数学
张飞	66	65	-
关羽	95	85	98
赵云	95	92	96
黄忠	90	88	77
典韦	80	90	90
典韦	80	90	90

数据存在 2 个问题：

- 典韦出现了 2 次
- 张飞的数学成绩缺失

缺失值处理办法

对于各种类型数据的缺失，我们到底要如何处理呢？以下是处理缺失值的四种方法：**删除记录，数据填补，和不处理。**

删除记录

- 简单粗暴
- 在样本数据量十分大且缺失值不多的情况下非常有效，但如果样本量本身不大且缺失也不少，那么不建议使用

```
"""
删除缺失数据
"""
import  pandas as pd
import  numpy as np

# 创建一个带有缺失值的DataFrame对象
df = pd.DataFrame(np.arange(12).reshape((3, 4)), index=['user1', 'user2',
'user3'], columns=['views', 'likes', 'transfers', 'saves'])
print("源数据: \n", df)
df.loc[:2, :1]= np.nan
print("缺失值数据: \n", df)

# 删除行，使用参数axis=0
print(df.dropna(axis=0))

# 删除列，  使用参数axis=1
print(df.dropna(axis=1))

# 删除数据表中含有空值的行
print(df.dropna())
```

数据填补

对缺失值的插补大体可分为两种：**替换缺失值**，**拟合缺失值，虚拟变量。**替换是通过数据中非缺失数据的相似性来填补，其核心思想是发现相同群体的共同特征，拟合是通过其他特征建模来填补，虚拟变量是衍生的新变量代替缺失值。

- **均值插补。**
    - **对于定类数据：**使用 **众数（mode）**填补，比如一个学校的男生和女生的数量，男生500人，女生50人，那么对于其余的缺失值我们会用人数较多的男生来填补。
        - **对于定量（定比）数据：**使用**平均数（mean）或中位数（median）**填补，比如一个班级学生的身高特征，对于一些同学缺失的身高值就可以使用全班同学身高的平均值或中位数来填补。

```
"""
均值差补
"""
import numpy as np
import  pandas as pd
df = pd.DataFrame(np.arange(12).reshape((4, 3)),
                  index=['user1', 'user2', 'user3', 'user4'],
                  columns=['price', 'count', 'info'])
df['price'][:1] = np.nan
df['price'][3] = 3

print("缺失值数据: \n", df)
# 使用price均值对NA进行填充
df_mean = df['price'].fillna(df['price'].mean())
# 使用price中位数对NA进行填充
df_median = df['price'].fillna(df['price'].median())
# 使用price众数对NA进行填充
df_mode = df['price'].fillna(df['price'].mode())

print("均值填充: \n", df_mean)
print("中位数填充: \n", df_median)
```

```
print("众数填充: \n", df_median)
```

- 插值模型
  - 拉格朗日插值法（`scipy`实现）

拉格朗日插值法可以找到一个多项式，其恰好在各个观测的点取到观测到的值。这样的多项式称为拉格朗日（插值）多项式。数学上来说，拉格朗日插值法可以给出一个恰好穿过二维平面上若干个已知点的多项式函数。

例如：当n=4时，上面的公式可简化为：

$$f(x)=\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}y_0+\frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}y_1+\frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}y_2+\frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}y_3$$

实例：

假设一个2次多项行式f, 按照前面的介绍，取三个点为

$$f(3)=10, f(6)=8, f(9)=4$$

$$l_0(3)=\frac{(x-6)(x-9)}{(3-6)(3-9)}$$
$$l_1(6)=\frac{(x-3)(x-9)}{(6-3)(6-9)}$$
$$l_2(9)=\frac{(x-3)(x-6)}{(9-3)(9-6)}$$

$$L(x)=f(3)l_0(3)+f(6)l_1(6)+f(9)l_2(9)$$
$$=10\frac{(x-6)(x-9)}{(3-6)(3-9)}+8\frac{(x-3)(x-9)}{(6-3)(6-9)}+4\frac{(x-3)(x-6)}{(9-3)(9-6)}$$
$$=\frac{-x^2+3x+90}{9}$$

插值$f(10)=\frac{20}{9}$

```
from scipy.interpolate  import lagrange
x = [3, 6, 9]
y = [10, 8, 4]
lagrange(x,y)
#poly1d([ -0.11111111,   0.33333333,  10.        ])

# 如果U7>:*? (tyyyyyyyyyyyy(要进行插值操作，可以：
lagrange(x, y)(10)
# 2.222222
```

  - 牛顿插值法（`scipy`未实现）

牛顿插值法是曲线拟合插值法中的一种，适合采用所有的数据都精确的情况下。

数据集成

多数据源数据仓库化。

- 实体识别
  - 同名异义
  - 异名同义
  - 单位不统一
- 冗余属性识别
  - 同一属性多次出现
  - 同一属性命名不一致导致重复

数据变换

- 规范化处理数据，便于使用。
  - 简单函数变换
    - 常用来将不具有正态分布的数据变换成具有正态分布的数据。
  - 规范化
    - 最小-最大规范化
    - 零-均值规范化（使用最多）
    - 小数定标规范化
  - 连续属性离散化
    - 等宽法
    - 等频法
    - 聚类
  - 属性构造
    - 推导属性
  - 小波变换
    - 新型数据分析工具

数据规约

- 降低错误数据对建模的影响，减少存储成本
  - 属性规约
  - 数值规约
    - 直方图
    - 聚类
    - 抽样
    - 参数回归

数据挖掘建模

根据挖掘目标和数据形式的不同，可以建立分类与预测、聚类分析、关联规则、时序模式、偏差检测等模型。

分类与预测

如餐饮行业想知道哪些用户会离我而去，哪些用户可能成为VIP。

- 常用的分类和预测算法:

算法名称	算法描述
回归分析	回归分析是确定预测属性（数值型）与其他变量间相互依赖的定量关系最常用的方法。包括线性回归、非线性回归、Logistic回归、岭回归、主成分回归、偏最小二乘回归等。
决策树	采用自顶两下的递归方式，在内部节点进行属性值的比较，并根据不同的属性值从该节点向下分支，最终得到的叶节点是学习划分的类。
人工神经网络	人工神经网络是一种模仿大脑神经网络的结构和功能而建立的信息处理系统，表示神经网络的输入和输出变量之间关系的模型。
贝叶斯网络	贝叶斯网络又称为信度网络，是Bayes方法的扩展， 是目前不确定知识表达和推理领域最有效的理论模型之一。
支持向量机	支持向量机是一种通过某种非线性映射，把低维的非线性可分转化为高维的线性可分，在高维空间进行线性分析的算法。

- python常用分类预测模型

模型	特点	来源
逻辑回归	基础的线性分类模型，简单有效	`sklearn.linear_model`
`SVM`	强大的模型，用来回归、预测、分类等，选取不同的核函数，模型线性或者非线性	`sklearn.svm`
决策树	基于“分类讨论，逐步细化”思想的分类模型，直观，易解释，图形化	`skearn.tree`
随机森林	思想类似决策树，精度更高，但是由于其随机性，丧失了决策树的可解释性	`sklearn.ensemble`
朴素贝叶斯	基于概率思想的简单有效的分类模型，给出容易理解的概率解释	`sklearn.naive_bayes`
神经网络	强大的拟合能力，用于拟合、分类等，很多强化版本如递神经网络，卷积神经网络，自编码器等，这些是深度学习的模型基础	`Keras`

聚类分析

- 群体上的考究。
- 和分类不同，这里所有的数据都是事先没有类别的。

- 这是非监督的学习算法。
- 原理：给定一组数据，根据数据自身距离或者相似度将其划分为若干组，原则为组内距离最小而组间距离最大。
- 常用聚类方法。
  - 划分方法（分裂）：K-Means算法（K-平均）、`K-MEDOIDS`算法（K-中心点）、`CLARANS`算法（基于选择的算法）
  - 层次分析方法：BIRCH算法（平均迭代规约和聚类）、CURE算法（代表点聚类）、CHAMELEON算法（动态模型）
  - 基于密度的方法：`DBSCAN`算法（基于高密度连接区域）、`DENCLUE算法`（密度分布函数）、OPTICS算法（对象排序识别）
  - 基于网格的方法：STING算法（统计信息网络）、`CLIOUE`算法（聚类高维空间）、WAVE-CLUSTER算法（小波变换）
  - 基于模型的方法：统计学方法、神经网络方法 - 常用聚类分析算法

算法名称	**算法描述**
K-Means	K-均值聚类也叫作快速聚类法，在最小化误差函数的基础上将数据划分为预定的函数K。原理简单，适合处理大量数据。
K-中心点	K-Means对于孤立点的敏感性，K-中心点不使用簇中对象的平均值作为簇中心，而选用簇中离平均值最近的对象作为簇中心。
系统聚类	也叫作多层次聚类，分类的单位由高到低呈现树形结构，所处位置越低，包含对象越少，但这些对象的共同特征越多。这种聚类方法只适合小数据量，会比较慢。