实习四 基于SQL实现机器学习的基本概念

成员：陈嘉2000013094 马佳媛2000013121 黄粤2000013095

In [180]:

```
%load_ext sql
import pymysql
pymysql.install_as_MySQLdb()
%sql mysql://stu2000013121:stu2000013121@162.105.146.37:43306
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

In [182]:

```
%sql show databases
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
4 rows affected.
```

Out[182]:

Database
dataset
information_schema
performance_schema
stu2000013121

In [183]:

```
%sql use dataset
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
0 rows affected.
```

Out[183]:

```
[]
```

In [184]:

```
%sql show tables;
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
18 rows affected.
```

Out[184]:

Tables_in_dataset
TheWorldHappinessReport2015
TheWorldHappinessReport2016
TheWorldHappinessReport2017
TheWorldHappinessReport2018
TheWorldHappinessReport2019
application
breast_cancer
diamonds
itemscore
movieGenres
movies
ratings
skyline_negative_correlation
skyline_positive_correlation
skyline_random
supermarketTrans
tags
xzqh

练习一：发现数据中隐含的辛普森悖论

辛普森悖论： 在分组比较中都占优势的一方，在总评中有时反而是失势的一方

In [5]:

```
%sql SELECT * FROM application;
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
240 rows affected.
```

Out[5]:

id	gender	department	acceptance
1	female	business	yes
10	female	business	yes
100	female	business	no
101	male	business	yes
102	male	business	yes
103	male	business	yes
104	male	business	yes
105	male	business	yes
106	male	business	yes

任务一：生成报表

In [6]:

```
%%sql
select (case
WHEN department ='business' then '商学院'
WHEN department ='law' then '法学院'
WHEN GROUPING(department) = 1 THEN '总计'
end )as 学院,
sum(IF(gender = 'female',1,0))as 女生申请,
sum(IF(gender = 'female'&&acceptance = 'yes',1,0))as 女生录取,
concat(round(sum(IF(gender = 'female'&&acceptance = 'yes',1,0))/sum(IF(gender = 'female',1,0))*
sum(IF(gender = 'male',1,0))as 男生申请,
sum(IF(gender = 'male'&&acceptance = 'yes',1,0))as 男生录取,
concat(round(sum(IF(gender = 'male'&&acceptance = 'yes',1,0))/sum(IF(gender = 'male',1,0))*100),
count(id) as 合计申请,
sum(IF(acceptance = 'yes',1,0))as 合计录取,
concat(round(sum(IF(acceptance = 'yes',1,0))/count(id)*100,1),'%') as 合计录取率
from application
group by department with rollup
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
3 rows affected.
```

Out[6]:

学院	女生申请	女生录取	女生录取率	男生申请	男生录取	男生录取率	合计申请	合计录取	合计录取率
商学院	100	49	49%	20	15	75%	120	64	53.3%
法学院	20	1	5%	100	10	10%	120	11	9.2%
总计	120	50	42%	120	25	21%	240	75	31.3%

由上表可知，商学院与法学院的男生录取率均高于女生录取率，但在总计中女生录取率高于男生录取率。在分组比较中男生都占优势，但在总评中反而是失势的一方，因此本录取信息中存在辛普森悖论。

任务二：使用SQL寻找一般性的辛普森悖论存在的方法

概率计算如下：

In [7]:

```
%%sql
select
concat(round(sum(IF(gender = 'male'&&acceptance = 'yes',1,0))/sum(IF(gender = 'male',1,0)),2)) a
concat(round(sum(IF(gender = 'female'&&acceptance = 'yes',1,0))/sum(IF(gender = 'female',1,0)),2
concat(round(sum(IF(gender = 'male'&&acceptance = 'no',1,0))/sum(IF(gender = 'male',1,0)),2)) as
concat(round(sum(IF(gender = 'female'&&acceptance = 'no',1,0))/sum(IF(gender = 'female',1,0)),2)
concat(round(sum(IF(gender = 'male'&&acceptance = 'yes'&&department = 'business',1,0))/sum(IF(g
concat(round(sum(IF(gender = 'female'&&acceptance = 'yes'&&department = 'business',1,0))/sum(IF
concat(round(sum(IF(gender = 'male'&&acceptance = 'yes'&&department = 'law',1,0))/sum(IF(gender
concat(round(sum(IF(gender = 'female'&&acceptance = 'yes'&&department = 'law',1,0))/sum(IF(gend
concat(round(sum(IF(gender = 'male'&&acceptance = 'no'&&department = 'business',1,0))/sum(IF(ge
concat(round(sum(IF(gender = 'female'&&acceptance = 'no'&&department = 'business',1,0))/sum(IF(
concat(round(sum(IF(gender = 'male'&&acceptance = 'no'&&department = 'law',1,0))/sum(IF(gender
concat(round(sum(IF(gender = 'female'&&acceptance = 'no'&&department = 'law',1,0))/sum(IF(gende

from application
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
1 rows affected.
```

Out[7]:

P(yes|男生)	P(yes|女生)	P(no|男生)	P(no|女生)	P(yes|<男生,商学院>)	P(yes|<女生,商学院>)	P(yes|<男生,法学院>)	P(yes|<女生,法学院>)	P(no|<男生,商学院>)	P(no|<女生,商学院>)	P(no|<男生,法学院>
0.21	0.42	0.79	0.58	0.75	0.49	0.10	0.05	0.25	0.51	0.90

寻找符合辛普森悖论的条件概率:

由上表可知:

P(yes|<男生，商学院>) > P(yes|<女生，商学院>) ∧
P(yes|<男生，法学院>) > P(yes|<女生，法学院>) ∧
P(yes|男生) < P(yes|女生)
⟹
P(no|<女生，商学院>)/P(yes|<女生，商学院>) > P(no|<男生，商学院>)/P(yes|<男生，商学院>)
∧
P(no|<女生，法学院>)/P(yes|<女生，法学院>) > P(no|<男生，法学院>)/P(yes|<男生，法学院>)
∧
P(no|女生)/P(yes|女生) < P(no|男生)/P(yes|男生)

因此，这些条件概率符合辛普森悖论

练习二 KNN分类

本次实习选择**威斯康辛乳腺癌数据集**进行机器学习的训练

任务一 属性值的预处理

In [8]:

```
%sql select * from breast_cancer;
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
699 rows affected.
```

Out[8]:

id	sample_number	thickness	cell_size	cell_shape	marginal_adhesion	single_epithelial_cell_size	bare
1	1000025	5	1	1	1	2	
2	1002945	5	4	4	5	7	
3	1015425	3	1	1	1	2	
4	1016277	6	8	8	1	3	
5	1017023	4	1	1	3	2	
6	1017122	8	10	10	8	7	
7	1018099	1	1	1	1	2	
8	1018561	2	1	2	1	2	

考虑到bare_nuclei一项中有取值为'?'的特殊行，可能会对后续距离的计算产生影响，因此在进行转换时将这些含有特殊符号的行省略

属性重命名

为方便后续的读写，在筛选可用行数据的同时，对属性值进行重命名操作

In [9]:

```
%%sql
drop table if exists stu2000013121.new_cancer;
create table stu2000013121.new_cancer as
    (select sample_number as 编号, thickness as 厚度,
    cell_size as 大小, cell_shape as 形状, marginal_adhesion as 边缘,
    single_epithelial_cell_size as 单核大小, bare_nuclei as 裸核,
    bland_chromatin as 染色质, normal_nucleoli as 核心, mitoses as 有丝分裂, class as 分类
    from breast_cancer
    where bare_nuclei!='?')
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
0 rows affected.
683 rows affected.
```

Out[9]:

```
[]
```

In [185]:

```
%%sql
use stu2000013121;
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
0 rows affected.
```

Out[185]:

[]

In [186]:

```
%%sql
show tables;
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
7 rows affected.
```

Out[186]:

Tables_in_stu2000013121
acc_K
distance
knn_cnt
knn_pre
new_cancer
test_data
train_data

In [12]:

```
%%sql
select * from new_cancer
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
683 rows affected.
```

Out[12]:

编号	厚度	大小	形状	边缘	单核大小	裸核	染色质	核心	有丝分裂	分类
1000025	5	1	1	1	2	1	3	1	1	2
1002945	5	4	4	5	7	10	3	2	1	2
1015425	3	1	1	1	2	2	3	1	1	2
1016277	6	8	8	1	3	4	3	7	1	2
1017023	4	1	1	3	2	1	3	1	1	2
1017122	8	10	10	8	7	10	9	7	1	4
1018099	1	1	1	1	2	10	3	1	1	2
1018561	2	1	2	1	2	1	3	1	1	2
1033078	2	1	1	1	2	1	1	1	5	2

经过统计后发现，class这一属性项刚好为二值分类，因此将class作为数据标签进行分类，其余属性均作为训练变量

In [13]:

```
%%sql
select 分类 from new_cancer
group by 分类
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
2 rows affected.
```

Out[13]:

分类
2
4

由于属性均为数值型属性，并不需要进行热值编码操作，
因此只对厚度、大小、形状、边缘、单核大小、裸核、染色质、核心、有丝分裂属性进行归一化操作

Min-Max归一化（Min-Max Normalization）

也称为离差标准化，是对原始数据的线性变换，使结果值映射到[0 – 1]之间。转换函数如下：

$$x^* = \frac{x - min}{max - min}$$

其中max为样本数据的最大值，min为样本数据的最小值。在每次实验中，每个属性的最大值均为10，最小值均为1

本实习中MinMaxScaler将通过估计器分别缩放和转换每个元素成[0,1]范围的值。

In [14]:

```
%%sql
update new_cancer set
        厚度 = (厚度-1)/(10-1),
        大小 = (大小-1)/(10-1),
        形状 = (形状-1)/(10-1),
        边缘 = (边缘-1)/(10-1),
        单核大小 = (单核大小-1)/(10-1),
        裸核 = (裸核-1)/(10-1),
        染色质 = (染色质-1)/(10-1),
        核心 = (核心-1)/(10-1),
        有丝分裂 = (有丝分裂-1)/(10-1)
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
683 rows affected.
```

Out[14]:

[]

In [15]:

```
%%sql
select * from new_cancer
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
683 rows affected.
```

Out[15]:

编号	厚度	大小	形状	边缘	单核大小	裸核	染色质	核
1000025	0.44444444	0	0	0	0.11111111	0	0.22222222	
1002945	0.44444444	0.33333333	0.33333333	0.44444444	0.66666667	1	0.22222222	0.111111
1015425	0.22222222	0	0	0	0.11111111	0.11111111	0.22222222	
1016277	0.55555556	0.77777778	0.77777778	0	0.22222222	0.33333333	0.22222222	0.666666
1017023	0.33333333	0	0	0.22222222	0.11111111	0	0.22222222	
1017122	0.77777778	1	1	0.77777778	0.66666667	1	0.88888889	0.666666
1018099	0	0	0	0	0.11111111	1	0.22222222	

任务二 数据集划分

随机数划分

插入区间为[0,1]的随机数，若随机数大于0.7则作为测试组，否则作为训练组

In [16]:

```
%%sql
alter table new_cancer add 随机数 float not NULL;
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
0 rows affected.
```

Out[16]:

```
[]
```

In [17]:

```
%%sql
update new_cancer set 随机数 = rand();
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
683 rows affected.
```

Out[17]:

```
[]
```

In [18]:

```
%%sql
select * from new_cancer
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
683 rows affected.
```

Out[18]:

编号	厚度	大小	形状	边缘	单核大小	裸核	染色质	核
1000025	0.44444444	0	0	0	0.11111111	0	0.22222222	
1002945	0.44444444	0.33333333	0.33333333	0.44444444	0.66666667	1	0.22222222	0.111111
1015425	0.22222222	0	0	0	0.11111111	0.11111111	0.22222222	
1016277	0.55555556	0.77777778	0.77777778	0	0.22222222	0.33333333	0.22222222	0.666666
1017023	0.33333333	0	0	0.22222222	0.11111111	0	0.22222222	
1017122	0.77777778	1	1	0.77777778	0.66666667	1	0.88888889	0.666666
1018099	0	0	0	0	0.11111111	1	0.22222222	

In [19]:

```
%%sql
drop table if exists test_data;
create table test_data as
    (select * from new_cancer
     where 随机数 >= 0.7)
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
0 rows affected.
196 rows affected.
```

Out[19]:

```
[]
```

In [20]:

```
%%sql
drop table if exists train_data;
create table train_data as
    (select * from new_cancer
     where 随机数 < 0.7)
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
0 rows affected.
487 rows affected.
```

Out[20]:

```
[]
```

In [21]:

```
%%sql
select * from test_data
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
196 rows affected.
```

Out[21]:

编号	厚度	大小	形状	边缘	单核大小	裸核	染色质	核
1017023	0.33333333	0	0	0.22222222	0.11111111	0	0.22222222	
1033078	0.33333333	0.11111111	0	0	0.11111111	0	0.11111111	
1036172	0.11111111	0	0	0	0.11111111	0	0.11111111	
1043999	0	0	0	0	0.11111111	0.22222222	0.22222222	
1047630	0.66666667	0.33333333	0.55555556	0.33333333	0.55555556	0	0.33333333	0.2222222
1054590	0.66666667	0.22222222	0.11111111	1	0.44444444	1	0.44444444	0.3333333
1056784	0.22222222	0	0	0	0.11111111	0	0.11111111	

In [22]:

```
%%sql
select * from train_data
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
487 rows affected.
```

Out[22]:

编号	厚度	大小	形状	边缘	单核大小	裸核	染色质	核
1000025	0.44444444	0	0	0	0.11111111	0	0.22222222	
1002945	0.44444444	0.33333333	0.33333333	0.44444444	0.66666667	1	0.22222222	0.111111
1015425	0.22222222	0	0	0	0.11111111	0.11111111	0.22222222	
1016277	0.55555556	0.77777778	0.77777778	0	0.22222222	0.33333333	0.22222222	0.666666
1017122	0.77777778	1	1	0.77777778	0.66666667	1	0.88888889	0.666666
1018099	0	0	0	0	0.11111111	1	0.22222222	
1018561	0.11111111	0	0.11111111	0	0.11111111	0	0.22222222	

任务三 实现KNN算法

把测试集和训练集样本之间的距离计算出来保存到**distance表**中，先确定一个K，针对每个测试集中的样本，选取前K个最近的训练集样本，根据这K个样本的最多类标签数决定测试样本的类别。并给出正确率。本实习使用欧式距离。

欧式距离阐述如下：

若样本数据有n个特征，且两点A和B的坐标表示为：

$$A(x_{11}, x_{12}, x_{13}, \ldots, x_{1n})$$

$$B(x_{21}, x_{22}, x_{23}, \ldots, x_{2n})$$

则A和B两点之间的欧式距离公式如下：

$$d_{AB} = \sqrt{\sum_{k=1}^{n} (x_{1k} - x_{2k})^2}$$

In [27]:

```
%%sql
drop table if exists distance;
create table distance as(
    select test_data.编号 as 测试编号, test_data.分类 as 测试分类, train_data.编号 as 训练编号,
    sqrt(power(test_data.厚度-train_data.厚度,2)+power(test_data.大小-train_data.大小,2)
         +power(test_data.形状-train_data.形状,2)+power(test_data.边缘-train_data.边缘,2)
         +power(test_data.单核大小-train_data.单核大小,2)+power(test_data.裸核-train_data.裸核,2
         +power(test_data.染色质-train_data.染色质,2)+power(test_data.核心-train_data.核心,2)
         +power(test_data.有丝分裂-train_data.有丝分裂,2)) as 距离
    from train_data join test_data
    order by 测试编号,距离
);
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
0 rows affected.
95452 rows affected.
```

Out[27]:

[]

In [116]:

```
%%sql
select * from distance
limit 10;
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
10 rows affected.
```

Out[116]:

测试编号	测试分类	训练编号	训练分类	距离
167528	2	1236043	2	0.29397236495634194
167528	2	1313982	2	0.33333333000000004
167528	2	1181356	2	0.36851387095524607
167528	2	743348	2	0.40061681269922256
167528	2	1197080	2	0.44444444999999994
167528	2	1331405	2	0.44444444999999994
167528	2	1257648	2	0.4581228499856914
167528	2	431495	2	0.45812285241104767
167528	2	1371920	2	0.45812285241104767
167528	2	1197979	2	0.45812285241104767

对于每一个测试编号，取前K个距离的分类值进行统计计数

In [187]:

```
%%sql
drop table if exists knn_cnt;
create table knn_cnt as(
    select 测试编号, 训练分类, count(*) as 分类数目
    from (select 测试编号, 训练分类 from
    (select 测试编号, 距离, 训练分类,
         @row_number := if(@test = 测试编号, @row_number + 1, 1) as row_num,
         @test:=测试编号 as test
        from distance)
    as tmp where row_num <= 17) as a
    group by 测试编号,训练分类
);

select * from knn_cnt;
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
0 rows affected.
231 rows affected.
231 rows affected.
```

Out[187]:

测试编号	训练分类	分类数目
167528	2	17
183913	2	17
191250	4	17
274137	4	17
303213	4	17
390840	4	17
428903	2	14

通过取最值从统计分类中选出较多的分类作为预测分类结果，存入**knn_pre**中

In [188]:

```
%%sql
drop table if exists knn_pre;
create table knn_pre as(
    select knn_cnt.测试编号, knn_cnt.训练分类 as 预测分类 from
        knn_cnt join
        (select 测试编号, max(分类数目) as 分类数
         from knn_cnt group by 测试编号) as knn_max
        on knn_cnt.测试编号 = knn_max.测试编号
        where knn_cnt.分类数目 = knn_max.分类数
);

delete from knn_pre where 测试编号 in (
    select * from (
        select 测试编号 from knn_pre
        group by 测试编号
        having count(测试编号) > 1
    ) tmp )
    and 预测分类 != 2;  #若分类为2和4的数目相同则取2作为分类

select * from knn_pre;
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
0 rows affected.
194 rows affected.
0 rows affected.
194 rows affected.
```

Out[188]:

测试编号	预测分类
167528	2
183913	2
191250	4
274137	4
303213	4
390840	4
428903	2

针对该K值计算本次的准确率

In [189]:

```
%%sql
select sum(acc)/count(*) 准确率 from
    (select test_data.编号, if(test_data.分类=knn_pre.预测分类, 1, 0) acc
     from test_data join knn_pre
     on test_data.编号 = knn_pre.测试编号) as tmp;
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
1 rows affected.
```

Out[189]:

准确率
0.9643

任务四 讨论最佳K

算法参数是k，从1~487更改K的值，观察准确率的变化

In [52]:

```
%%sql
#drop table if exists acc_K;
create table acc_K(
    K int primary key,
    accuracy double
);
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
0 rows affected.
0 rows affected.
```

Out[52]:

[]

In [190]:

```
%%sql
insert into acc_K
select 17, sum(acc)/count(*) as 准确率 from
    (select test_data.编号, if(test_data.分类=knn_pre.预测分类, 1, 0) acc
     from test_data join knn_pre
     on test_data.编号 = knn_pre.测试编号) as tmp;

select * from acc_K
order by K;
```

```
 * mysql://stu2000013121:***@162.105.146.37:43306
1 rows affected.
27 rows affected.
```

Out[190]:

K	accuracy
1	0.964285714
2	0.948979591
3	0.969387755
4	0.954081632
5	0.964285714
6	0.959183673
7	0.964285714
8	0.959183673

对上表分析发现，K取奇数的准确率普遍高于K取偶数时的准确率。

分析原因：这与统计最近距离点时的数值筛选有关，偶数的K个点可能出现两种分类一样多的情况，在这种情况下，人为判给了分类2，由此可能导致误差，而奇数的K值则避免了这种情况。

当K值大于200时，准确率出现了明显的下降.

K值与准确率的变化趋势符合如下结论：

K值越大，模型的偏差越大，对噪声数据越不敏感；
K值很大时，可能造成模型欠拟合；
K值越小，模型的方差就会越大；
K值太小，容易过拟合。

与scikit-learn包获得的准确率作对比

为不影响数据集的分组，使用相同的训练集和测试集进行对比

In [170]:

```
import pandas as pd
import pymysql
from sklearn.model_selection import train_test_split

#连接数据库，拉取数据表
connection = pymysql.connect(host='162.105.146.37',
                             port=43306,
                             user='stu2000013121',
                             password='stu2000013121',
                             database='stu2000013121')
cursor = connection.cursor()

#从训练集和测试集分别拉取数据转化为dataframe进行训练
cursor.execute("select * from train_data")
data1 = cursor.fetchall()
frame1 = pd.DataFrame(data1)
train_data = frame1

cursor.execute("select * from test_data")
data2 = cursor.fetchall()
frame2 = pd.DataFrame(data2)
test_data = frame2

column = ['编号','厚度','大小','形状','边缘','单核大小','裸核','染色质','核心','有丝分裂','分类']
train_data.columns = column
test_data.columns = column
```

选择进行训练和要分类的属性

In [172]:

```
X_train = train_data.drop(['编号', '随机数', '分类'], axis=1)
y_train = train_data['分类']
X_test = test_data.drop(['编号', '随机数', '分类'], axis=1)
y_test = test_data['分类']
```

对每个测试点都有487个距离，使K在1~487之间变动，记录准确率的变化

同时统计最高准确率的数值与对应的K值

In [191]:

```
from sklearn.neighbors import KNeighborsClassifier

maxx = 0
tmp = 0
for k in range(1, 487):
    knn = KNeighborsClassifier(n_neighbors=k, weights='uniform')
    knn.fit(X_train, y_train)
    print(f"k = {k}, accuracy = {knn.score(X_test, y_test)}")
    if knn.score(X_test, y_test) > maxx:
        tmp = k
    maxx = max(maxx, knn.score(X_test, y_test))

print("max accuracy = ", maxx)
print("best k = ", tmp)
```

```
k = 469, accuracy = 0.6326530612244898
k = 470, accuracy = 0.6326530612244898
k = 471, accuracy = 0.6326530612244898
k = 472, accuracy = 0.6326530612244898
k = 473, accuracy = 0.6326530612244898
k = 474, accuracy = 0.6326530612244898
k = 475, accuracy = 0.6326530612244898
k = 476, accuracy = 0.6326530612244898
k = 477, accuracy = 0.6326530612244898
k = 478, accuracy = 0.6326530612244898
k = 479, accuracy = 0.6326530612244898
k = 480, accuracy = 0.6326530612244898
k = 481, accuracy = 0.6326530612244898
k = 482, accuracy = 0.6326530612244898
k = 483, accuracy = 0.6326530612244898
k = 484, accuracy = 0.6326530612244898
k = 485, accuracy = 0.6326530612244898
k = 486, accuracy = 0.6326530612244898
max accuracy =  0.9693877551020408
best k =  3
```

对比上述结果可知，

sklearn的最小的最佳K值为3，对应的最高准确率为0.9694

sql得到的最小的最佳K值为3，对应的最高准确率为0.9694，与sklearn结果一致，因此取得了较好的效果