openGauss AI特性创新实践课



华为技术有限公司

# 关卡一、openGauss数据安装及基本操作

openGauss数据安装及基本操作, 作业提交任务如下：

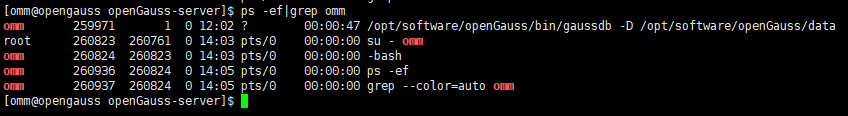
任务一：数据库状态验证

1. 查询数据库状态成功截图



任务二：数据库服务进程验证

1. 查看数据库服务进程截图（包含数据库服务器的主机名）



安装数据库所需要的步骤：

首先服务器需要有第三方编译库，若无则先下载。然后下载openGauss的源码并存放至指定目录。然后通过文件传输工具将cmake包上传至服务器并解压，同时还要安装依赖包和替换python的版本。

将用户切换到omm，设定omm的环境变量，并使其生效；然后进入openGauss的源码目录下生成配置文件；紧接着使用make命令编译，使用make install命令进行安装。最后初始化数据库。

实验思考题：为什么需要通过源码编译，安装数据库？

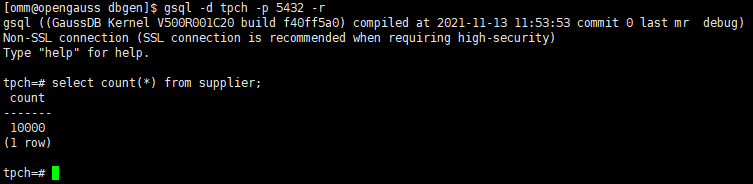
答：通过源码编译，可以自定义一些配置，自己配置一些安装的目录、cmake版本等；还要自己下载一些编译所需要的依赖包。通过源码编译可以知道自己在做什么，清楚安装一个数据库的过程都有哪些，遇到问题的时候能够找方法去解决，锻炼自己的能力。

# 关卡二、openGauss数据导入及基本操作

任务一：数据初始化验证

1. 查询supplier表的行数，并将结果进行图：

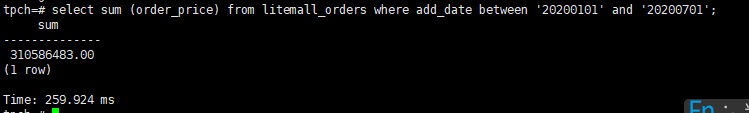
select count(\*) from supplier;;



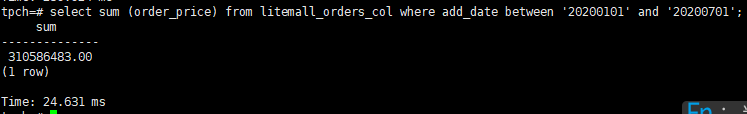
任务二：行存表与列存表执行效率对比

1. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的总和查询，并对比执行效率截图

select sum (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';

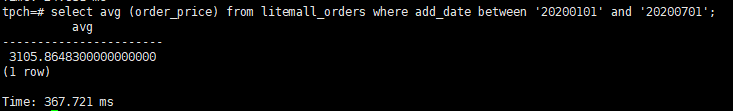


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

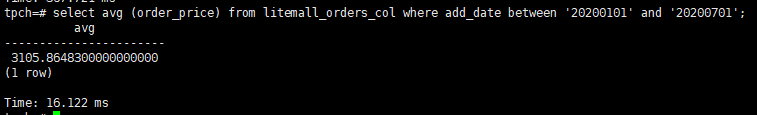


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

select avg (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';

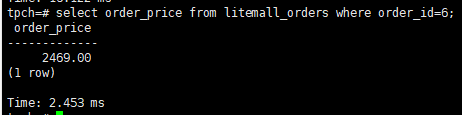


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

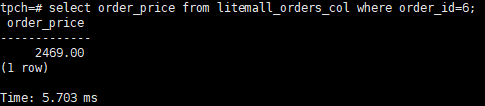


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

select order\_price from litemall\_orders where order\_id=6;



select order\_price from litemall\_orders\_col where order\_id=6;



4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

update litemall\_orders set order\_price=2468 where order\_id=6;



update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

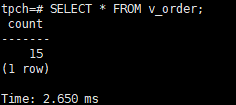
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



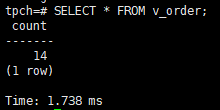
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;



5. 对表进行操作后，刷新增量物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM vi\_order;



实践思考题1：行存表与列存表在执行相同的SQL语句时，为何执行的时间不同？在执行哪些类型SQL时，行存表效率更高？在执行哪些类型SQL时，列存表效率更高？

1. 因为两者的存储结构不同，行存表中的数据是按行进行存储，而列存表的数据是以列进行存储，对于相同的SQL语句，行存表和列存表对数据的检索方式不同，所执行的时间也会不同。如在前两句聚合运算的SQL中，在列存表中只用查询一列，并对其进行计算；而在行存表中需要将所有列都查询到，再找到对应列进行计算，这大大增加了运行的时间。在第三句SQL中，列存表只用查找一列即可，行存表要搜索所有数据，再去除冗余列，增加了I/O操作，增加了时间。在更新操作中，行存表中可以通过order\_id快速定位到要更新的行从而更新对应的数据，在列存表中需要不断的查找列，才能定位到对应的字段更新数据，因此导致了行存表比列存表效率高。
2. 需要进行频繁的插入和更新操作、需要在行中查询所有属性的操作、需要随机的增删查改操作，这类型的SQL中在行存表的效率更高。
3. 在需要对数据进行聚合运算、表中有很多属性但只需查找少量属性的操作，这类型的SQL在列存表中的效率更高。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

在全量物化视图中，他的刷新机制是将所有数据重新查询加载一遍，需要的时间会比较长；二在增量物化视图中，刷新视图的时候是只将改变的数据重新加载，而不是将所有数据重新加载，大大节省了刷新视图的时间。

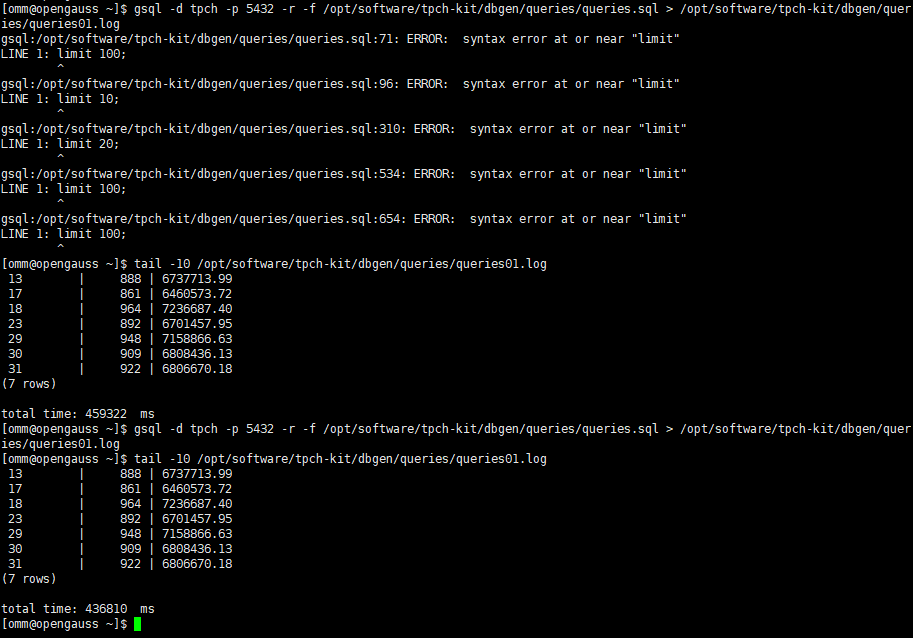
# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

1. 执行TPCH脚本，获得测试时间，将执行结果截图：

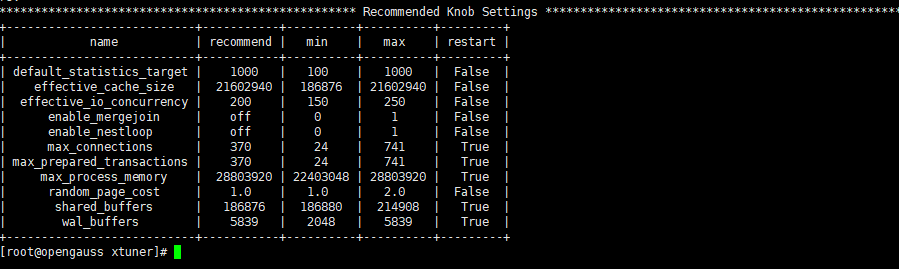
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log

运行了两次，第一次运行结束后发现有错误，修改完成后重新运行，并查看时间。



2. 使用root用户，执行X-Tuner进行参数建议优化，将执行结果截图

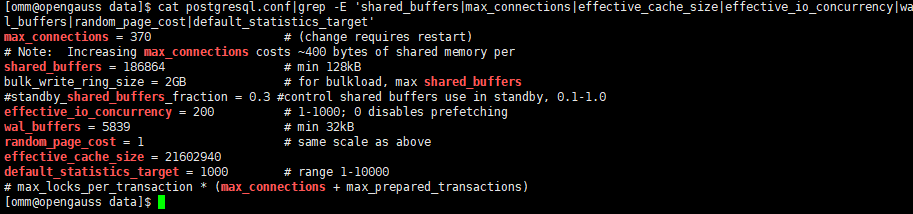
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



3.重启完成后，获取参数值：

cd /opt/software/openGauss/data

cat postgresql.conf|grep -E 'shared\_buffers|max\_connections|effective\_cache\_size|effective\_io\_concurrency|wal\_buffers|random\_page\_cost|default\_statistics\_target'



任务二：使用Index-advisor对select 查询语句进行优化，并通过对比执行计划，得到优化前后的不同。

1. 使用explain，对查询2020年3月订单表收入并进行排序的SQL加以分析，将结果截图。

EXPLAIN

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

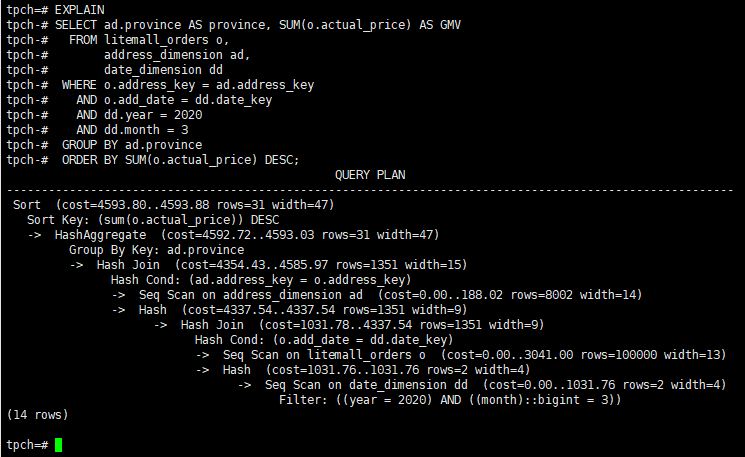
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC;



2. 使用索引推荐功能，对查询语句进行推荐，将执行结果截图。

select \* from gs\_index\_advise('

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

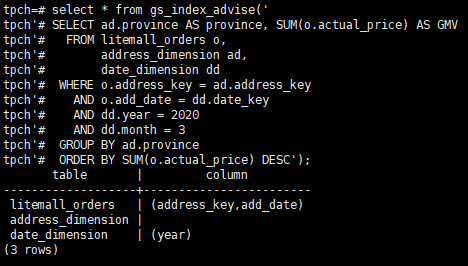
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

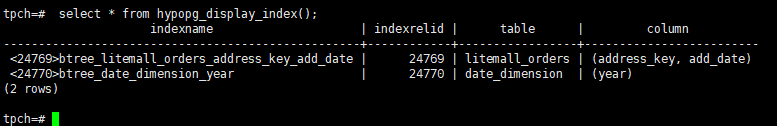
GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC');



3. 查看创建的虚拟索引列，将执行结果截图。

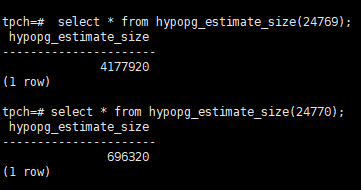
select \* from hypopg\_display\_index();



4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

select \* from hypopg\_estimate\_size(16715);

select \* from hypopg\_estimate\_size(16716);



5.再次使用explain，对该SQL加以分析，将执行结果截图。

EXPLAIN

SELECT ad.province AS province, SUM(o.actual\_price) AS GMV

FROM litemall\_orders o,

address\_dimension ad,

date\_dimension dd

WHERE o.address\_key = ad.address\_key

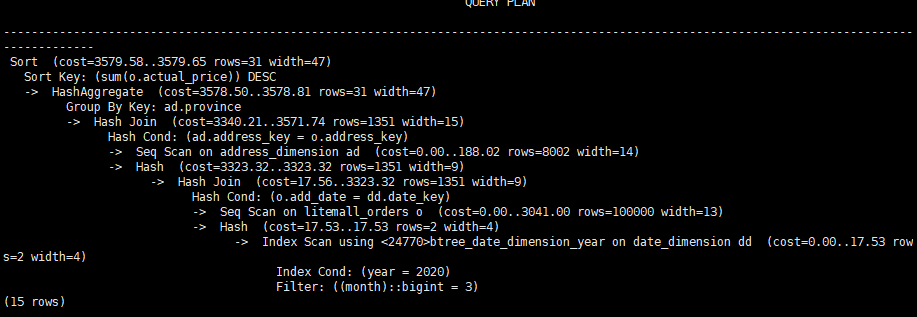
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

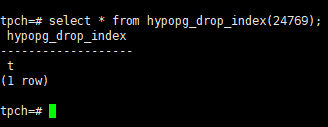
GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC;



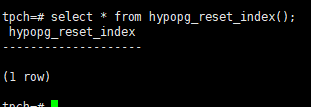
6. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_drop\_index(16715);



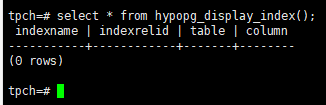
7. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_reset\_index();



8. 查看索引虚拟列，将执行结果截图。

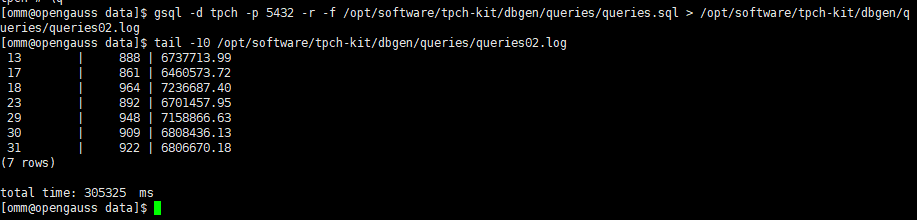
select \* from hypopg\_display\_index();



任务三：通过创建索引，对queries.sql中的SQL语句进行优化，并对比优化前后queries.sql执行的时间。

1. 重新执行queries.sql查询，将执行结果截图：

gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



挑战一：进一步优化queries.sql中的查询语句，使得前后执行时间出现倍数级的提升。

1. 重新执行queries.sql查询，将执行结果截图：

gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries03.log



实践思考题1：根据X-Tuner给出的参数优化，在哪些参数上进行了优化，为何要对这些参数进行优化？

shared\_buffers：用于缓存的专用内存量

max\_connections:系统支持的最大连接数

effective\_cache\_size：提供可用于磁盘高速缓存的内存量的估计值

effective\_io\_concurrency：同时被执行的并发磁盘 I/O 操作的数量

wal\_buffers：预写日志记录的缓冲区的大小

random\_page\_cost：规划器对一次非顺序获取磁盘页面的代价估计

default\_statistics\_target：采样颗粒度

对这些参数优化，可以最大限度的使用系统的性能。

实践思考题2：索引的使用，对于执行SQL有什么好处？除了使用索引和参数外，还有哪些方面可以对数据库进行优化？

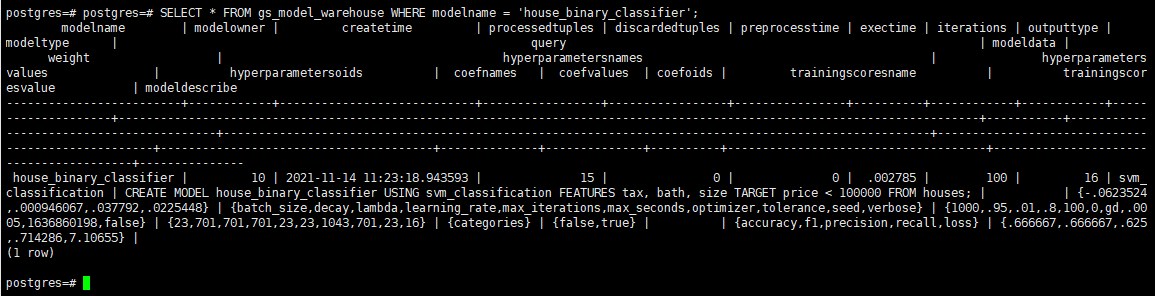
可以大大加快数据的检索速度；可以加速表和表之间的连接；在使用分组和排序字句进行检索数据是，可以减少查询中分组和排序的时间；在查询的过程中使用优化隐藏器，提高了系统的性能。

还可以分区、分表，进行并行查询。

# 关卡四、openGauss的DB4AI特性应用

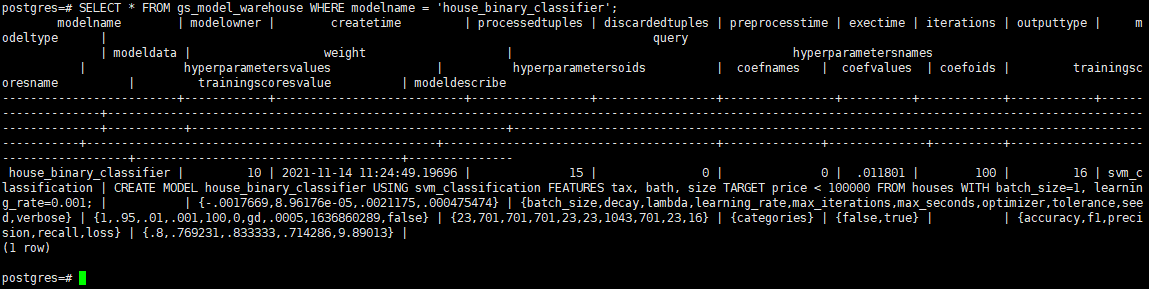
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';



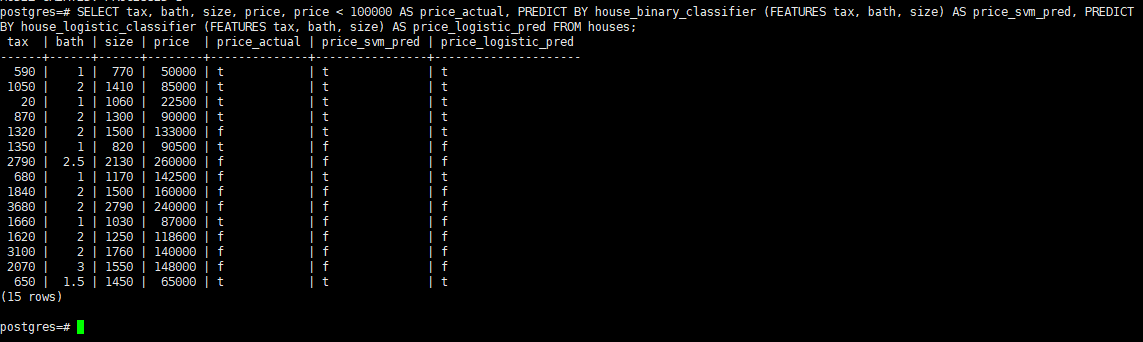
任务二：观察新模型的信息，将执行结果截图。

postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';



任务三：利用训练好的逻辑回归模型预测数据，并与SVM算法进行比较，将执行结果截图。

postgres=# SELECT tax, bath, size, price, price < 100000 AS price\_actual, PREDICT BY house\_binary\_classifier (FEATURES tax, bath, size) AS price\_svm\_pred, PREDICT BY house\_logistic\_classifier (FEATURES tax, bath, size) AS price\_logistic\_pred FROM houses;



实践思考题1：分类模型与回归模型有何不同？

两者的区别在于需要预测值的类型的不同，分类模型是对离散值进行预测；回归模型是对连续值进行预测；

实践思考题2：什么是SVM算法？

支持向量机（support vector machines，SVM）是一种二分类模型，它将实例的特征向量映射为空间中的一些点，从而利用这一些空间中的点进行分类，该算法使用于中小型数据样本、非线性、高维的分类问题。

实践思考题3：分类问题有哪些评价指标，请分别说明他们的含义？

准确率：对于给定的数据，分类正确的样本数占总样本数的比例

精确率：在被所有预测为正的样本中实际为正样本的概率

召回率：在实际为正的样本中被预测为正样本的概率

F1分数：精确率和召回率的调和平均

ROC曲线：横坐标为假正率(FPR)，纵坐标为真正率(TPR)，ROC曲线越陡，那么模型的性能也就越好

AUC：表示ROC中曲线下的面积，用于判断模型的优劣

实践思考题4：回归问题有哪些评价指标，请分别说明他们的含义？

平均绝对误差（Mean Absolute Error，MAE）：指预测值与真实值之间平均相差多大，能更好地反映预测值误差的实际情况

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，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

R-square(决定系数)： 分母理解为原始数据的离散程度，分子为预测数据和原始数据的误差，二者相除可以消除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。