openGauss AI特性创新实践课



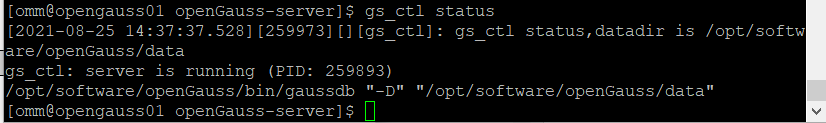
华为技术有限公司

# 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：

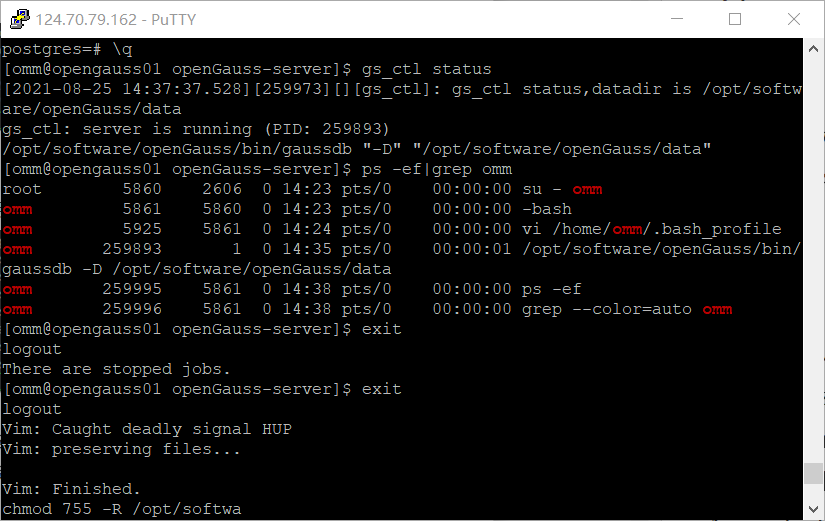
任务一：数据库状态验证

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）



实验思考题：为什么需要通过源码编译，安装数据库？

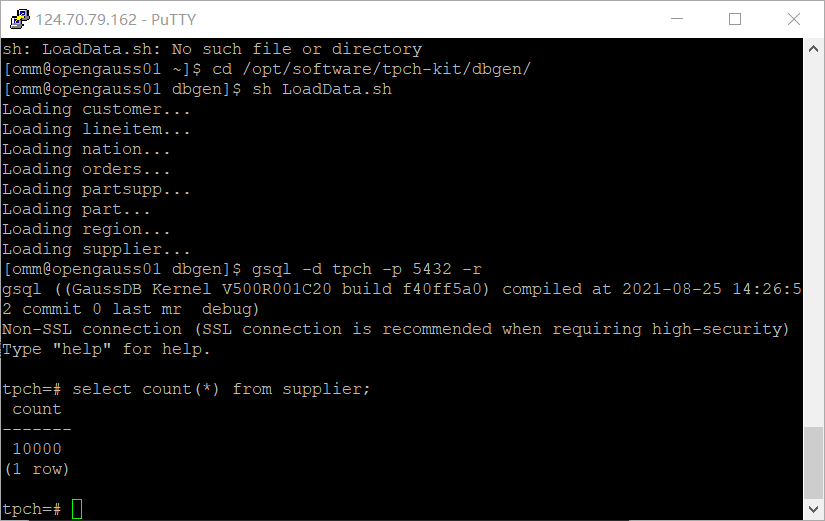
答：因为这样做可以安装指定的版本并指定编译参数，按照需求进行安装，灵活性比较大。

# 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：

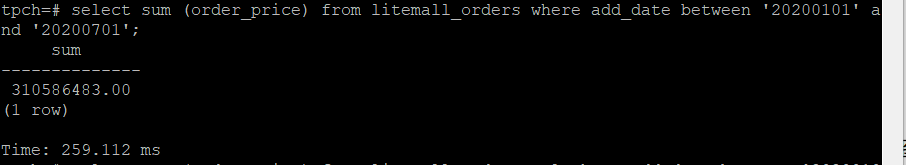
select count(\*) from supplier;;



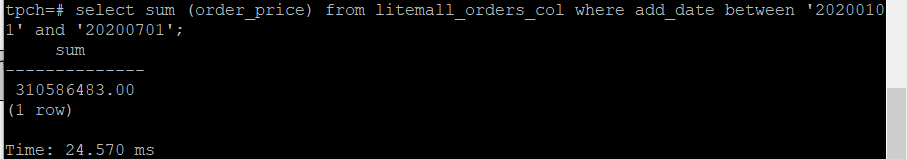
任务二：行存表与列存表执行效率对比

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';

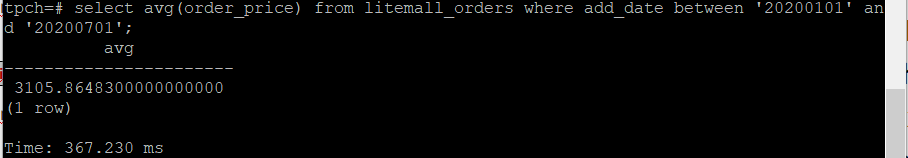


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

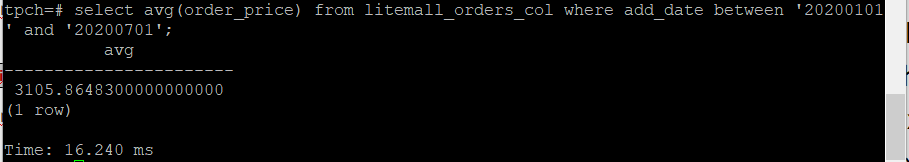


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

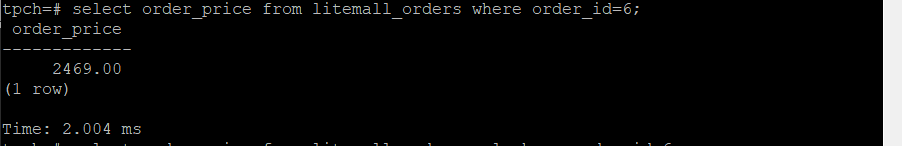


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

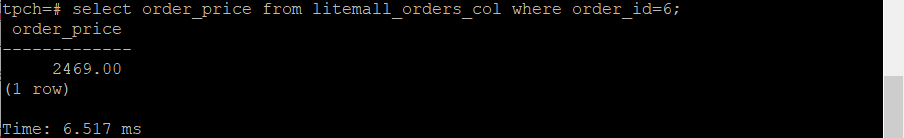


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

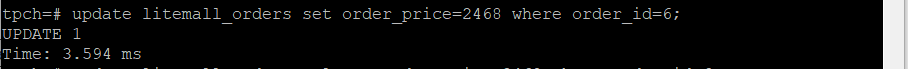


select order\_price from litemall\_orders\_col where order\_id=6;

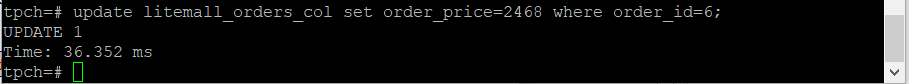


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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



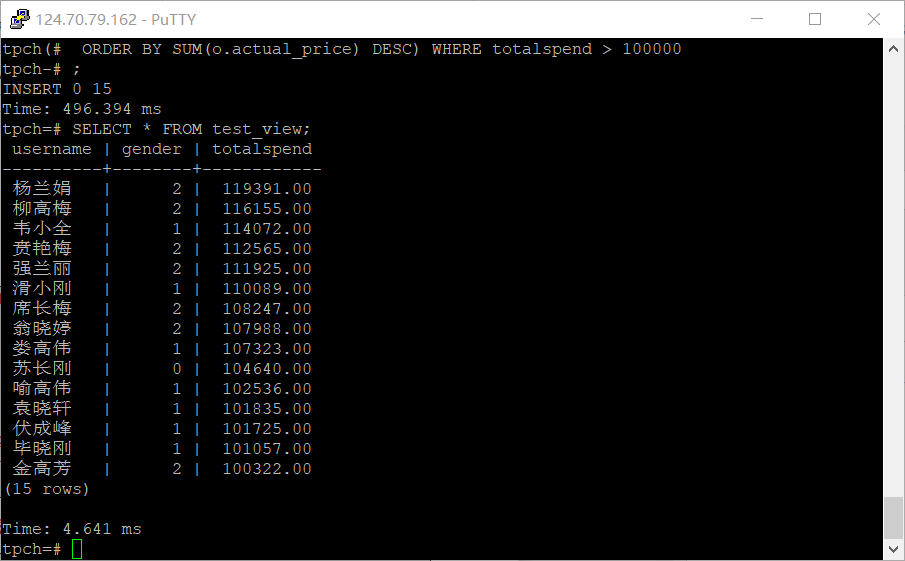
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



任务三：物化视图的使用

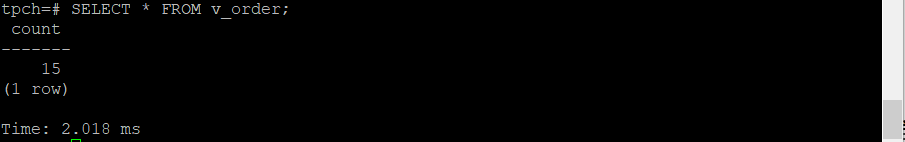
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



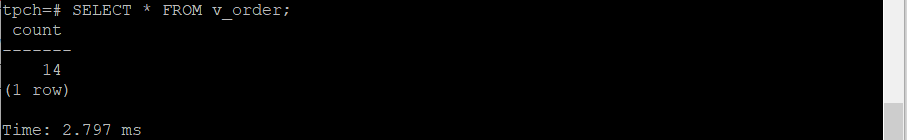
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



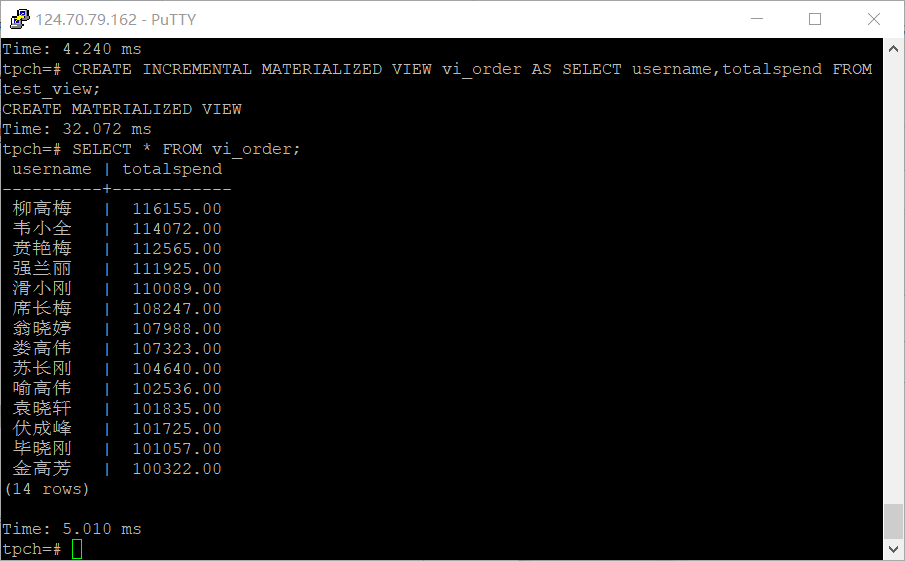
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



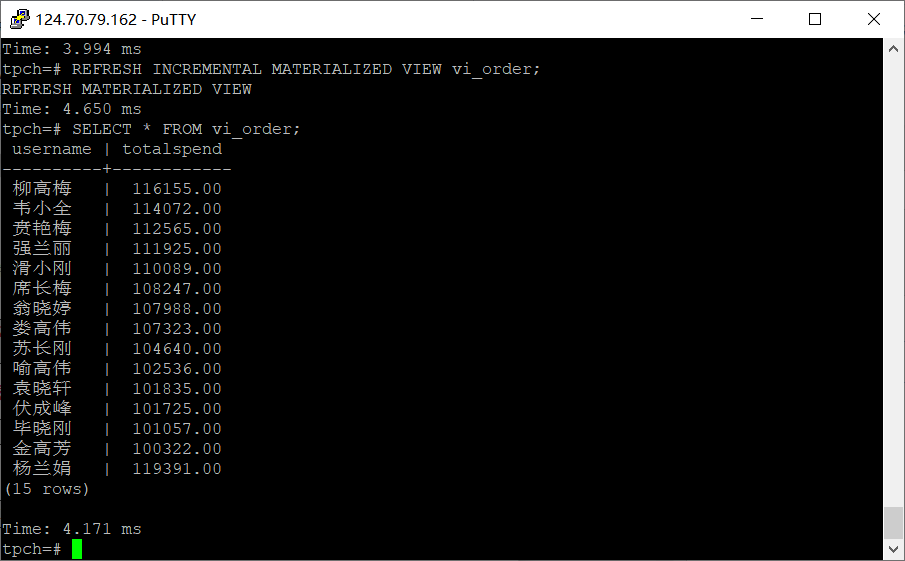
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

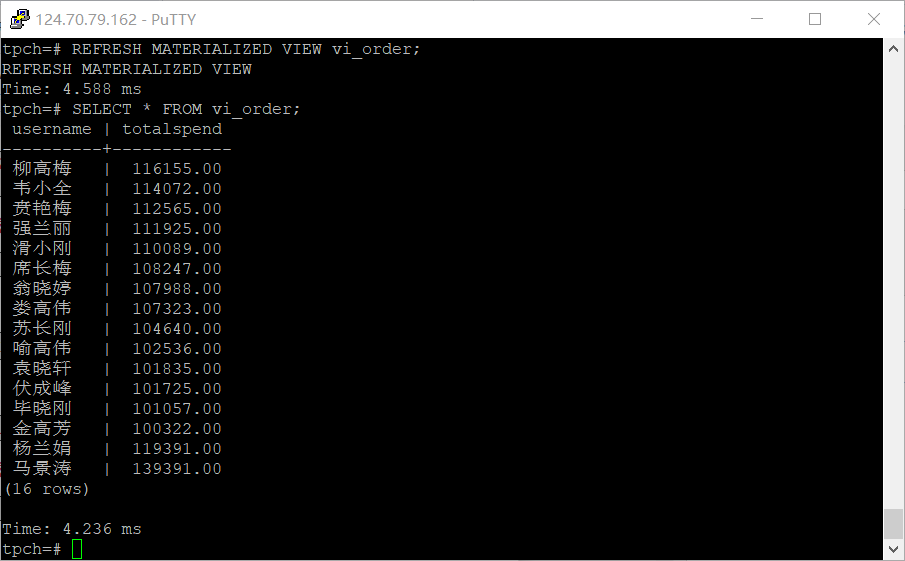
SELECT \* FROM vi\_order;



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？

答：因为行存是以行形式存储表中，一行数据存在一个数据文件中。适用于具有许多迭代事务的OLTP类型的工作负载以及一次需要多列的单行，因此检索是高效的；列存是以列为形式组织存储，每列对应一个或一批文件。若有一个宽表，在读取某一个列数据时，行存需要把匹配的列的所在的行数据块都要扫描一遍，而列存可以避免。

如果主要关注的是整张表的内容，而不是单独某几列，并且所关注的内容是不需要通过任何聚集运算的，那么行存表效率更高；如果注的都是某几列的内容，或者有频繁聚集需要的，通过聚集之后进行数据分析的表。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

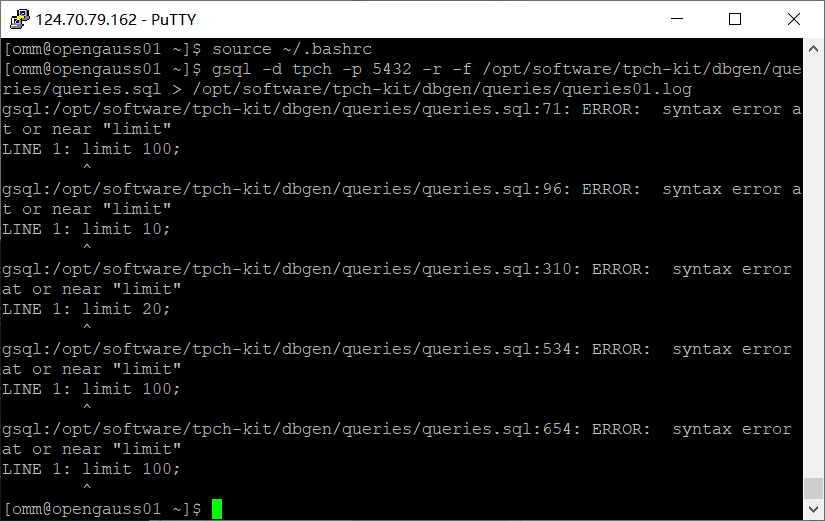
答：全量物化视图仅支持对创建好的物化视图做全量更新，而不支持做增量更新；而增量物化视图可以对物化视图增量刷新，但与全量创建物化视图相比所支持场景较小，目前物化视图创建语句仅支持基表扫描语句或者UNION ALL语句。

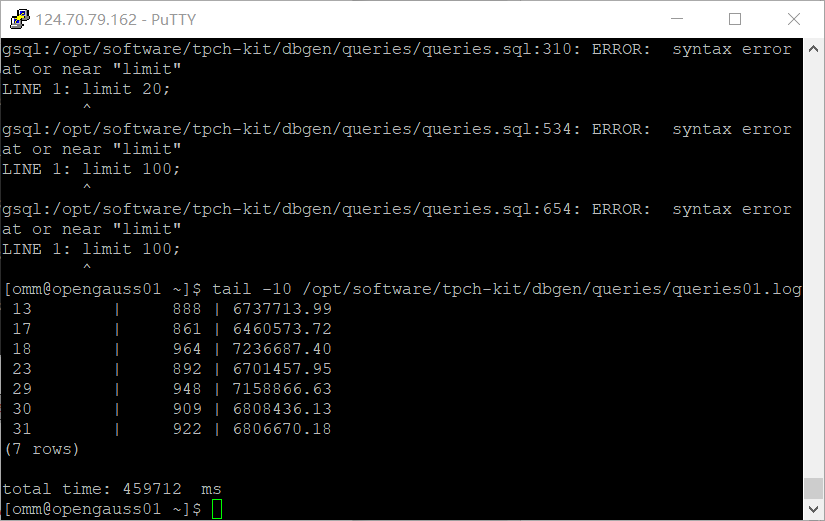
# 关卡三、openGauss的AI4DB特性应用

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：

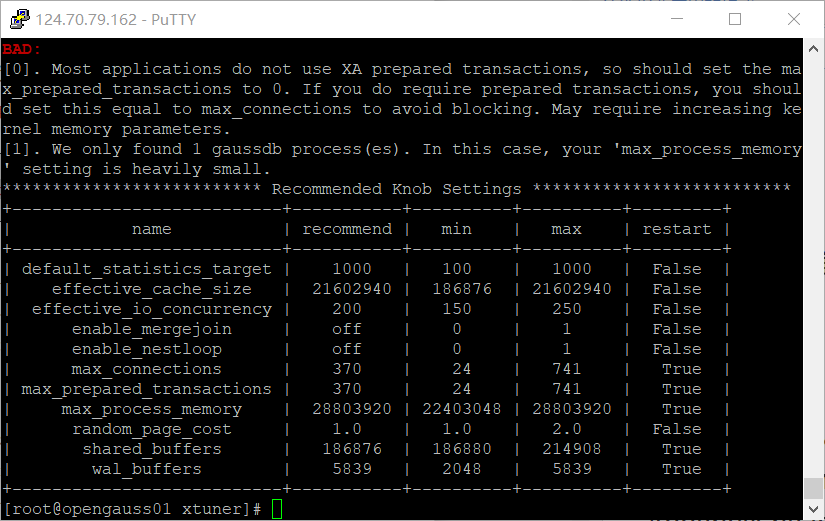
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log





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

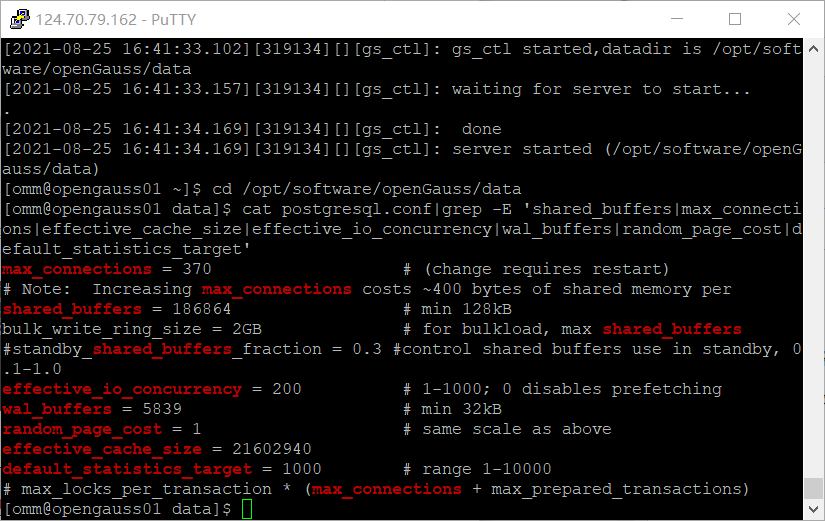
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

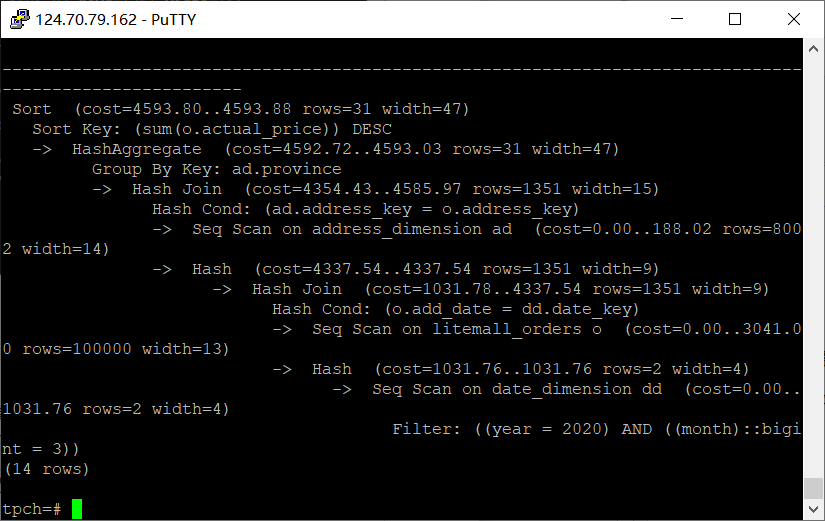
AND o.add\_date = dd.date\_key

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

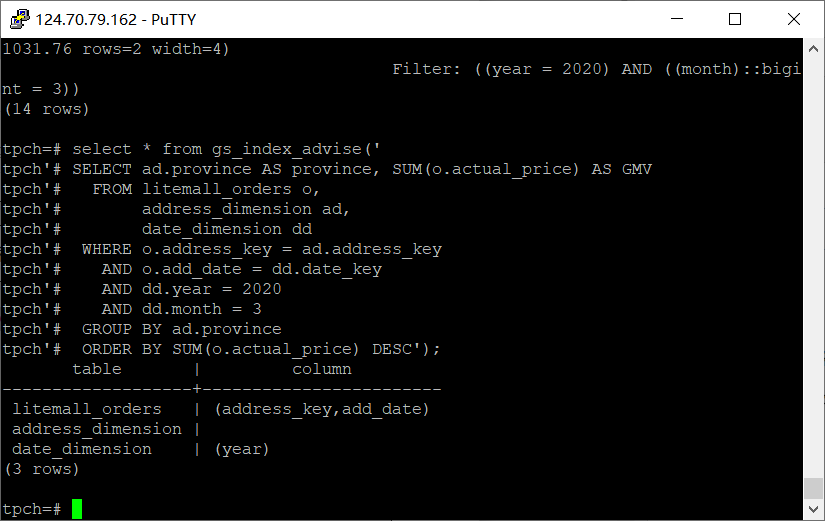
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

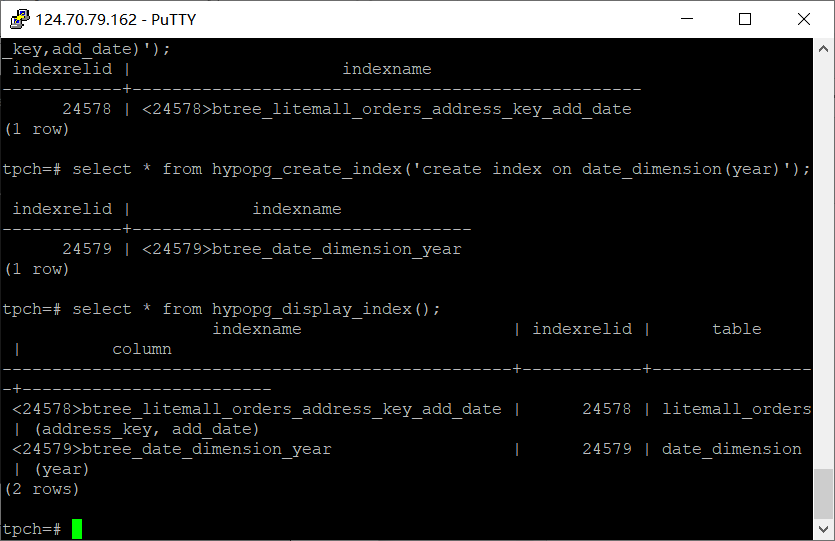
GROUP BY ad.province

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。

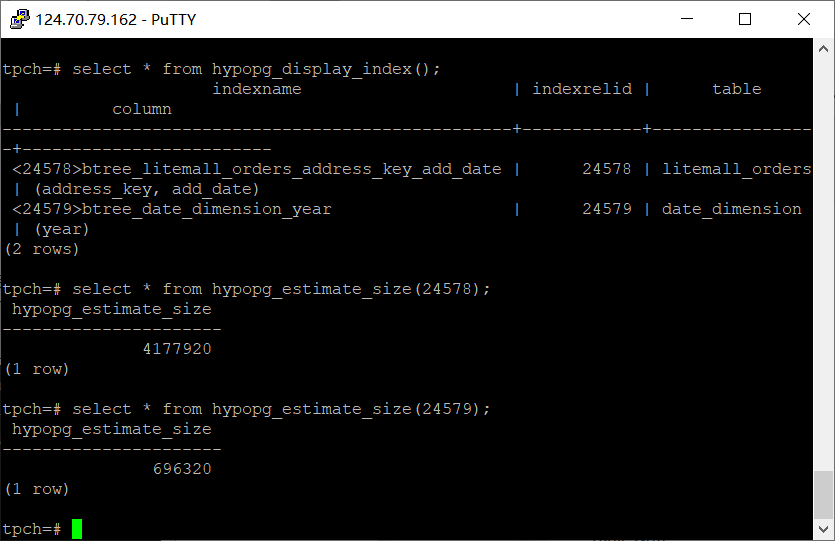
select \* from hypopg\_display\_index();



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

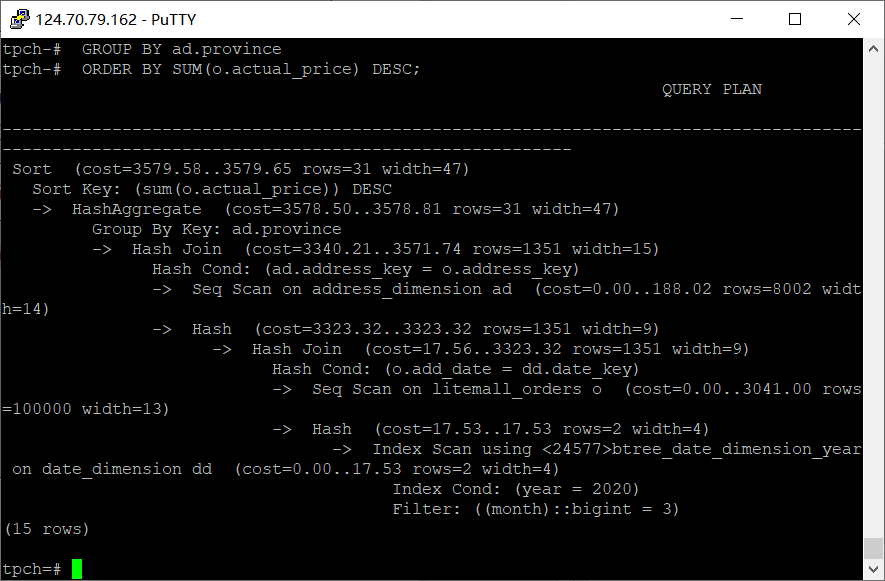
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

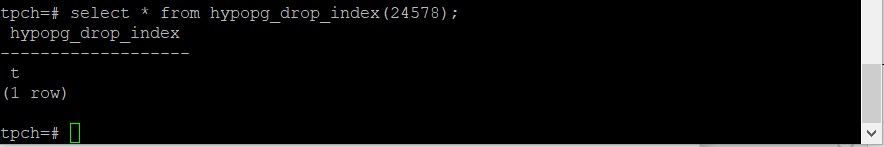
GROUP BY ad.province

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

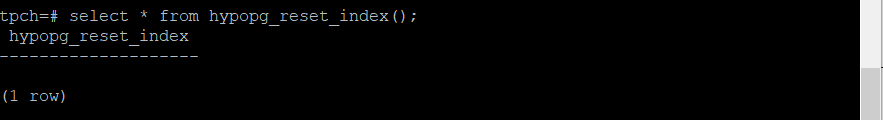


6. 删除某一个索引虚拟列，将执行结果截图。

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

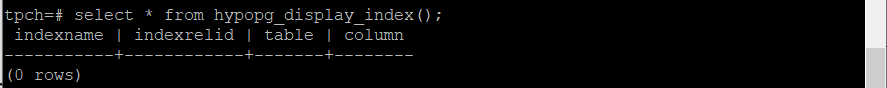
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

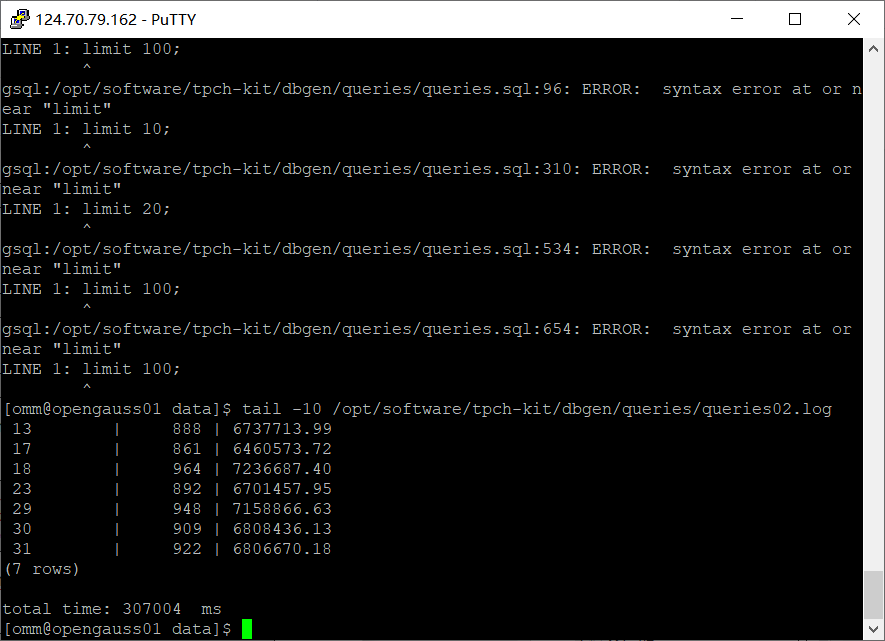
select \* from hypopg\_display\_index();



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？

答：对参数address\_key,add\_date进行了优化。因为可以建立索引，提高执行语句的效率。

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？

答：（1）创建索引可以大大提高系统的性能。

第一，通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。

第二，可以大大加快 数据的检索速度，这也是创建索引的最主要的原因。

第三，可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。

第四，在使用分组和排序 子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。

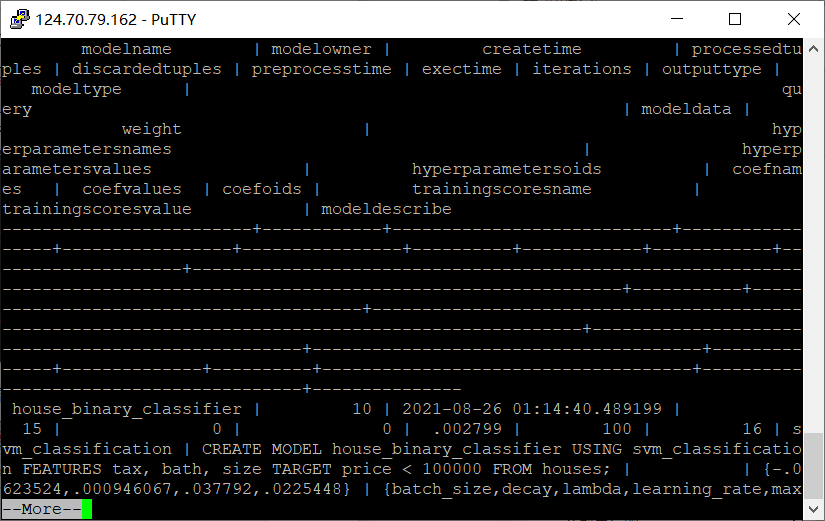
第五，通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

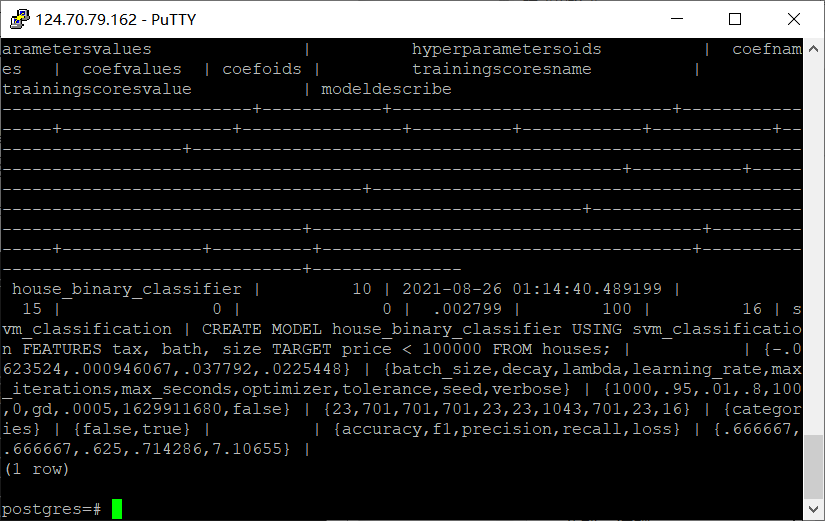
（2）除了使用索引和参数外，还可以通过选取最适用的字段属性、使用连接来代替子查询、使用联合来代替手动创建的临时表、事务、锁定表、优化的查询语句等对数据库进行优化。

# 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：

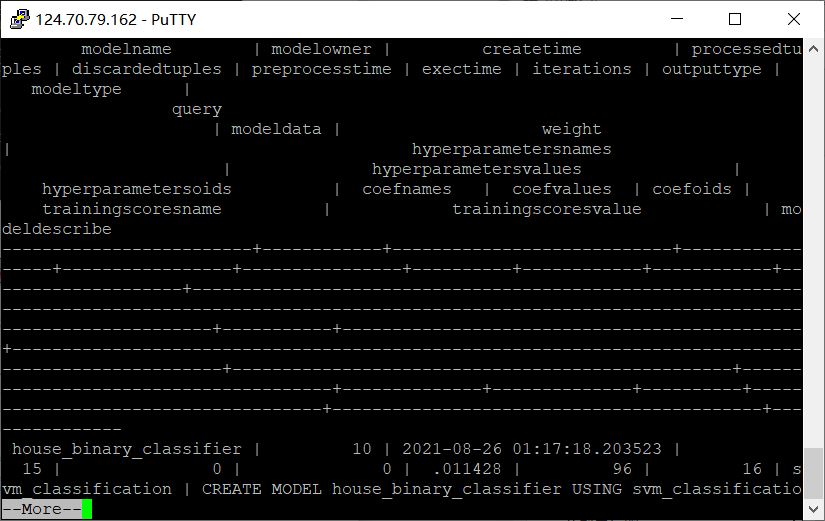
postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';

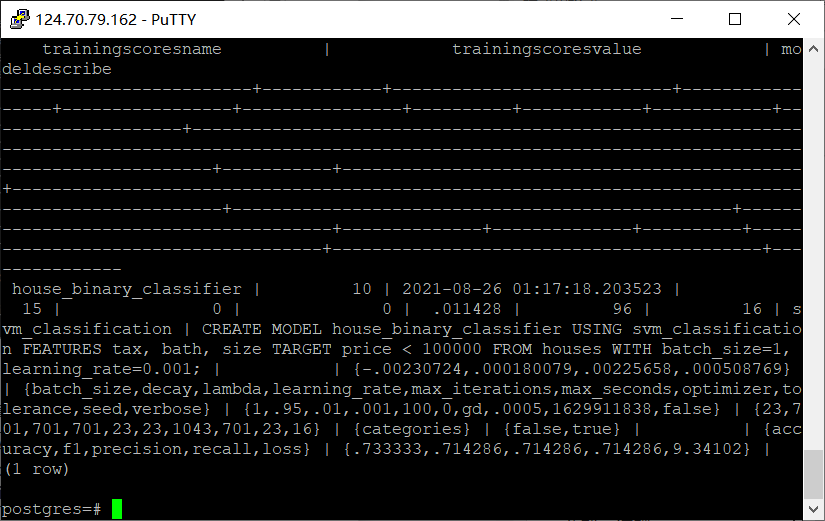




任务二：观察新模型的信息，将执行结果截图。

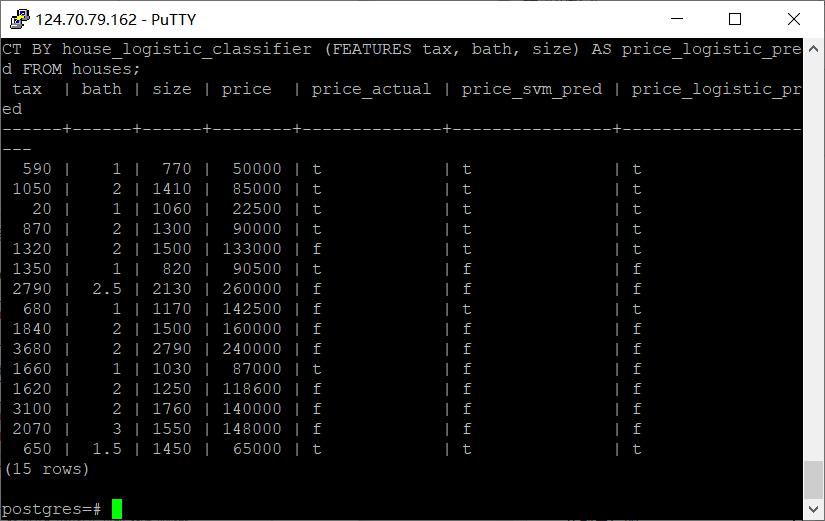
postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';





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？

答：分类模型是将回归模型的输出离散化。

实践思考题2：什么是SVM算法？

答：SVM是一个二元分类算法，线性分类和非线性分类都支持。SVM按监督学习方式对数据进行二元分类的广义线性分类器，其决策边界是对学习样本求解的最大边距超平面。SVM使用铰链损失函数计算经验风险并在求解系统中加入了正则化项以优化结构风险，是一个具有稀疏性和稳健性的分类器，可以通过核方法进行非线性分类，是常见的核学习方法之一。

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？

答：分类问题的评价指标有：准确率、精确率、召回率、F1-Score、AUC、ROC、P-R曲线、混淆矩阵。

准确率（Accuracy）：准确率=预测正确的样本数/所有样本数，即预测正确的样本比例（包括预测正确的正样本和预测正确的负样本）。

精确率（Precision）：精确率/查准率衡量的是所有被预测为正例的样本中有多少是真正例，是针对预测结果而言的一个评价指标。

召回率（Recall）：真正例率（TPR）/灵敏度（Sensitivity）/召回率（Recall）/查全率衡量的是所有的正例中有多少是被正确分类了，也可以看作是为了避免假负例(FN)的发生。是针对原始样本而言的一个评价指标。

F1-Score：能够很好的评估模型，其主要用于二分类问题。针对精准率和召回率都有其自己的缺点；如果阈值较高，那么精准率会高，但是会漏掉很多数据；如果阈值较低，召回率高，但是预测的会很不准确。本质上是一种调和平均数（与算术平均和几何平均相比，调和平均更注重小值）。

AUC（Area under ROC Curve）：AUC可以通过对ROC曲线下各部分的面积求和而得。

ROC曲线：接收者操作特征曲线（receiver operating characteristic curve），是反映敏感度和特异度（= 1 - 误诊率）连续变量的综合指标，ROC曲线上每个点反映着对同一信号刺激的感受性。在ROC曲线以TPR为纵、以FPR为横轴作图，显示出一种正例与负例之间的“博弈”。尤其是评估不平衡数据。

P-R曲线（Precision-Recall Curve）：当预测样本后（假设样本预测结果以 概率形式 输出），对样本结果进行排序，排在最前面（概率最大）的模型认为“最可能是”正例的样本，排在最后的是模型认为”最不可能“是正例的样本。按此顺序设置不同的阈值（阈值可以是排序后的概率值 / 固定划分点），在不同的阈值下，计算出当前阈值下的精确率P和召回率R。以精确率P为纵轴，召回率R为横作图，就可以得到P-R曲线。

混淆矩阵：混淆矩阵也称误差矩阵，是表示精度评价的一种标准格式，用n行n列的矩阵形式来表示。具体评价指标有总体精度、制图精度、用户精度等，这些精度指标从不同的侧面反映了图像分类的精度。

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？

答：回归问题的评价指标有：平均绝对误差、均方误差、决定系数、交叉验证。

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）：是观测值与真值偏差的平方和与观测次数的比值。

决定系数(R-square)： 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。“决定系数”是通过数据的变化来表征一个拟合的好坏。

交叉验证（Cross-Validation）：交叉验证的基本思想是把在某种意义下将原始数据(dataset)进行分组,一部分做为训练集(train set)，另一部分做为验证集(validation set or test set)。首先用训练集对分类器进行训练，再利用验证集来测试训练得到的模型(model)，以此来做为评价分类器的性能指标。