openGauss AI特性创新实践课



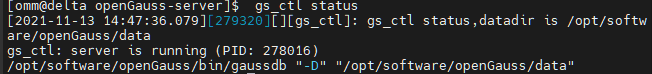
华为技术有限公司

# 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：

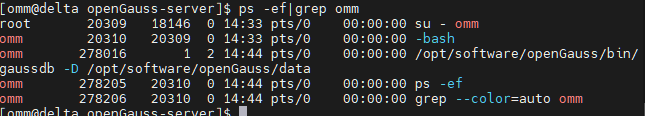
任务一：数据库状态验证

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）



主要步骤：

①创建omm用户并切换。

②下载opengauss解压第三方编译库和源码

③设置环境变量

④编译安装opengauss

⑤初始化与运行

实验思考题：为什么需要通过源码编译，安装数据库？

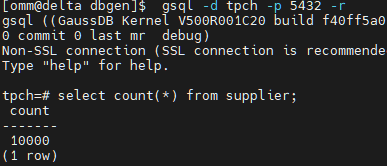
通过源码编译安装过程，可以设定参数，按照需求进行安装。并且安装的版本，可以自己选择，灵活性比较大。用于学习，可以更了解过程中的细节，需要什么环境和学习一些参数的含义。

# 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：

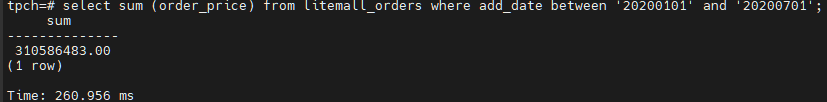
select count(\*) from supplier;



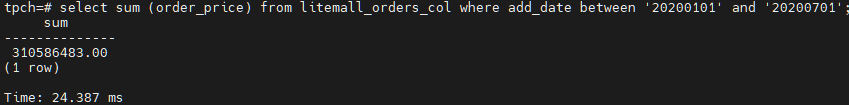
任务二：行存表与列存表执行效率对比

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';

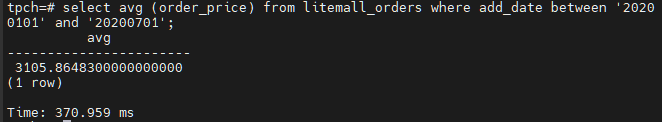


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

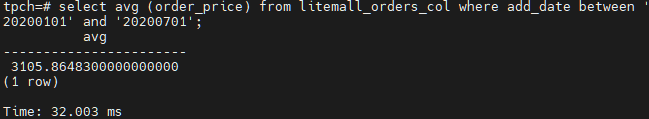


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

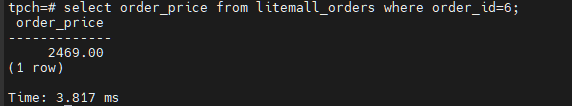


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

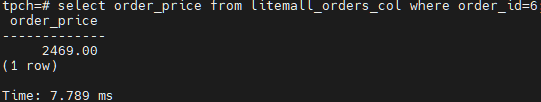


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

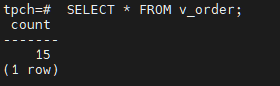
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



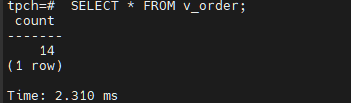
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

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;



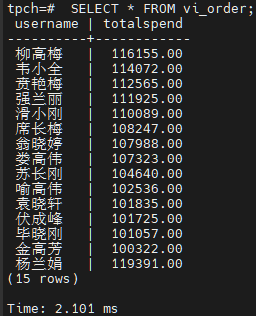
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

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？

行存表与列存表主要在于物理存储结构的不同，计算机读取数据一般以行为读取方式；

因此列存表把每个记录同一属性作为行时，SQL语句对这一属性的遍历操作会更快，但是要SQL操作涉及一条记录里的多个属性时，性能会快速下降，添加记录也会涉及物理存储的扩展问题。

行存储对单一记录操作很快，对属性遍历较慢，添加记录的比较简单。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

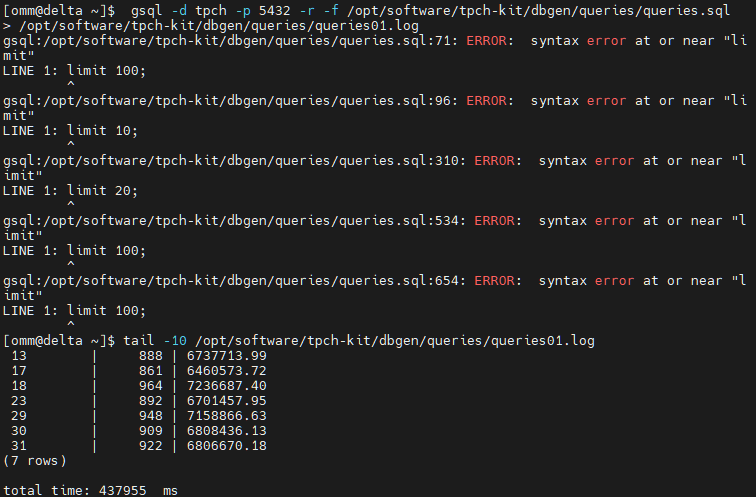
全量物化视图仅支持对创建好的物化视图做全量更新而不支持做增量更新。增量物化视图需要用户手动执行语句完成对物化视图在一段时间内的增量数据进行刷新，目前增量物化视图所支持场景较小。但实验中是对单表的操作，因此两种物化视图都可行。

# 关卡三、openGauss的AI4DB特性应用

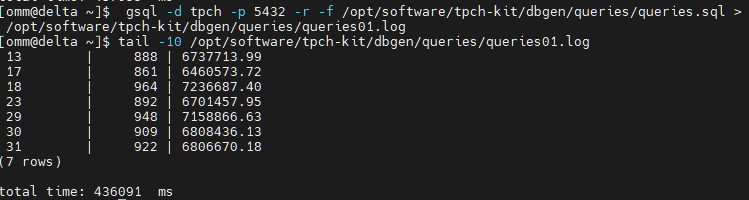
任务一：使用X-Tuner进行参数优化

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

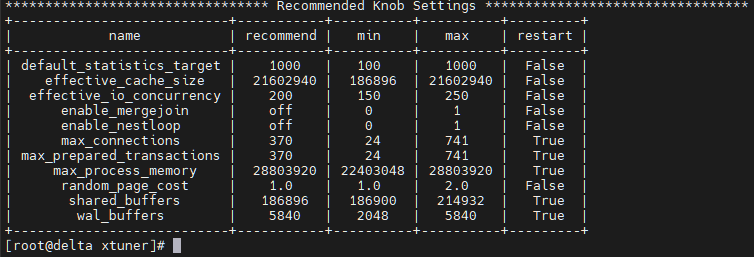


修改几处queries.sql中的语句后：



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

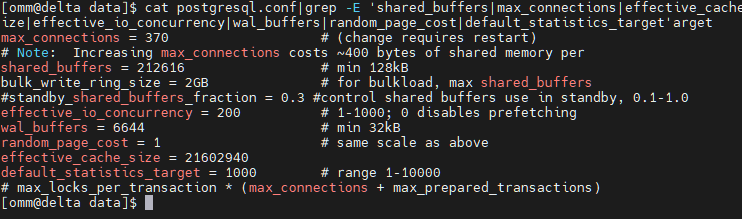
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

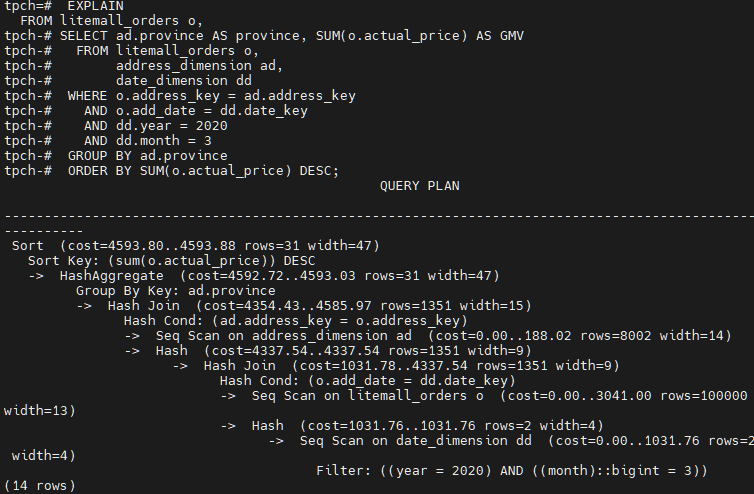
AND o.add\_date = dd.date\_key

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

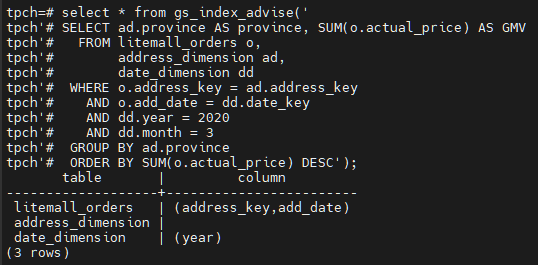
AND o.add\_date = dd.date\_key

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。

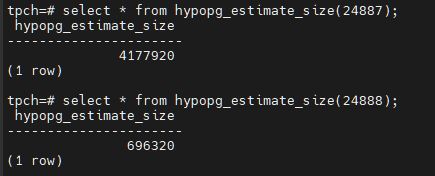
select \* from hypopg\_display\_index();



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

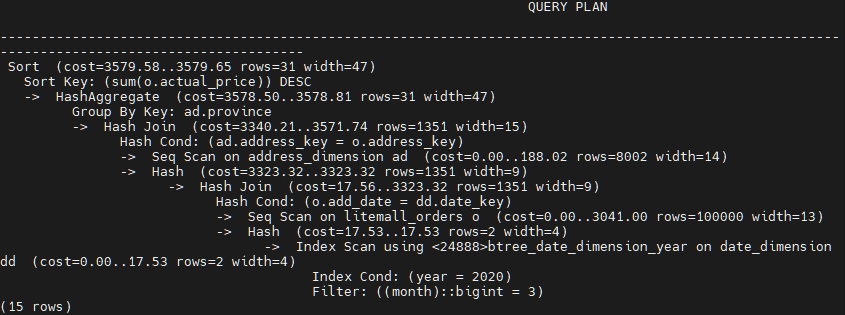
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

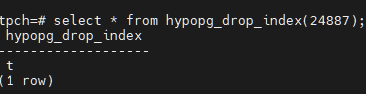
GROUP BY ad.province

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



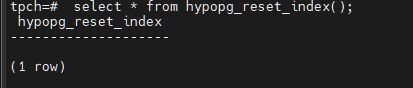
6. 删除某一个索引虚拟列，将执行结果截图。

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



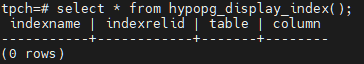
7. 删除所有索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

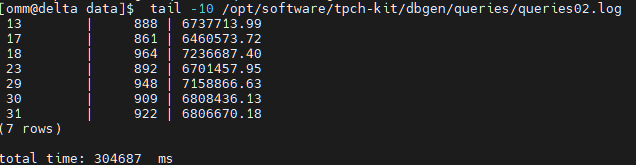
select \* from hypopg\_display\_index();



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？

对以下参数进行了优化：

shared\_buffers : 该参数可以调整缓冲区的大小。数据库在从磁盘中查询数据前，会先查找shared\_buffers的页，若命中则返回，可以避免磁盘查找。通常来说该值应该高于默认值来获得更好的性能。

max\_connections：最大连接数，应该根据服务器负载的上限进行设置。本次实验中没有很大连接数量要求，根据X-Tuner给出的参数优化即可。

effective\_cache\_size：系统能提供的cache大小总和，该参数会被用到使用索引的成本考虑之中，值越大，使用索引的可能性越大。

effective\_io\_concurrency ：控制I/O性能的参数，为0时禁止I/O并发查询。

wal\_buffers：WAL（预写日志）记录写入的缓冲区的大小，设置该值大一些可以在有大量并发连接的时候提供更好的性能。

random\_page\_cost ：用于决定使用索引的代价是否值得；如果系统Cache使用状况良好，那么random\_page\_cost可以适当降低。

default\_statistics\_target：告诉 openGauss 应该抽样多少数据来填充存储元数据的表。较大的值会增加ANALYZE所需的时间，但可能会提高计划者评估的质量

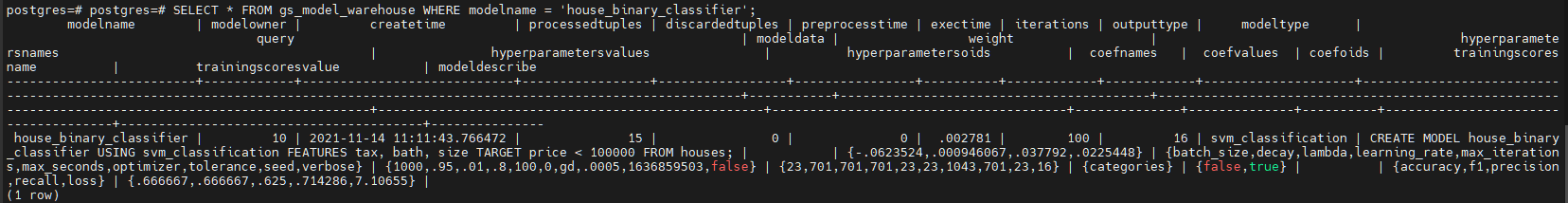
**实践思考题2：**索引的使用，对于执行SQL有什么好处？除了使用索引和参数外，还有哪些方面可以对数据库进行优化？

引索是用空间和一些维护所需性能来换取查询时的快速定位查询目标。还有SQL语句优化，用分区表，并行查询，压缩表。

# 关卡四、openGauss的DB4AI特性应用

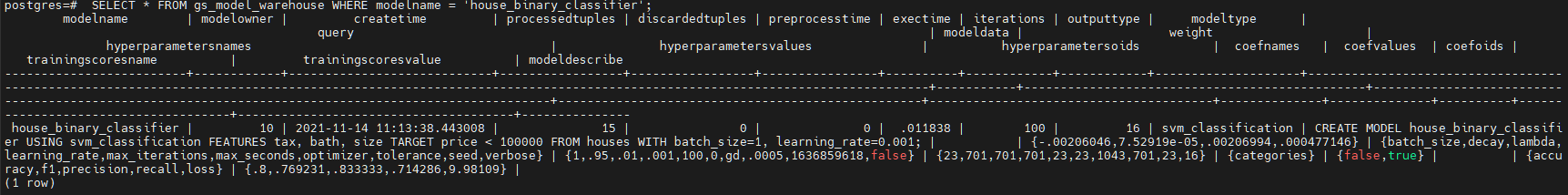
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



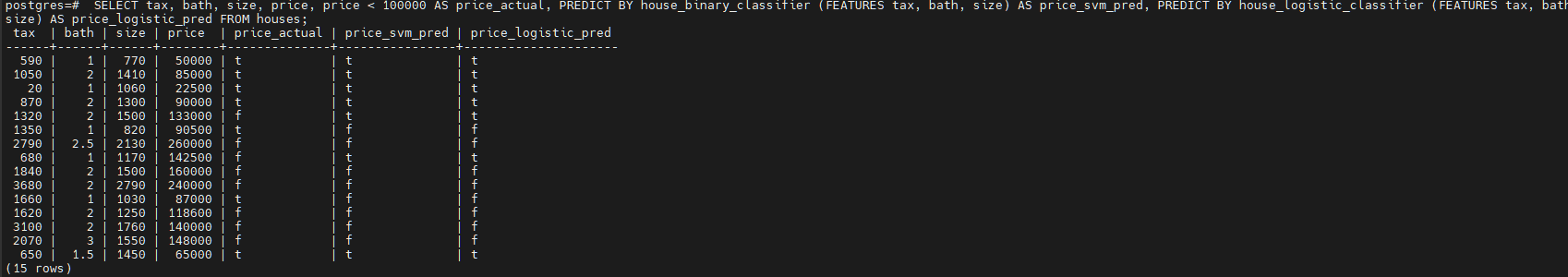
任务二：观察新模型的信息，将执行结果截图。

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？

定量输出称为回归，或者说是连续变量预测；  
 定性输出称为分类，或者说是离散变量预测。

实践思考题2：什么是SVM算法？

SVM是一个二元分类算法，线性分类和非线性分类都支持。

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(Accuracy)](https://blog.csdn.net/manduner/article/details/91040867" \l "3Accuracy_10" \t "https://blog.csdn.net/manduner/article/details/_self)：对于给定的测试数据集，分类器正确分类的样本数与总样本数之比。

[精确率(Precision)：检索出相关文档数与检索出的文档总数的比率（正确分类的正例个数占分类为正例的实例个数的比例），衡量的是检索系统的查准率。](https://blog.csdn.net/manduner/article/details/91040867" \l "4PrecisionRecallF1_14" \t "https://blog.csdn.net/manduner/article/details/_self)

[召回率(Recall)](https://blog.csdn.net/manduner/article/details/91040867" \l "4PrecisionRecallF1_14" \t "https://blog.csdn.net/manduner/article/details/_self)： 是指检索出的相关文档数和文档库中所有的相关文档数的比率（正确分类的正例个数占实际正例个数的比例），衡量的是检索系统的查全率。

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(MAE, Mean Absolute Error)

这个指标是对绝对误差损失的预期值.

平均绝对百分比误差(MAPE, Mean Absolute Percentage Error)

这个指标是对相对误差损失的预期值.所谓相对误差,就是绝对误差和真值的百分比.

均方误差(MSE, Mean Squared Error)

该指标对应于平方(二次)误差的期望.

均方误差根或均方根误差(RMSE, Root Mean Squared Error)

该指标对应于平方(二次)误差的期望.

均方误差对数(MSLE, Mean Squared Log Error)

该指标对应平方对数(二次)差的预期.

中位绝对误差(MedAE, Median Absolute Error)

通过取目标和预测之间的所有绝对差值的中值来计算损失.

R Squared又叫可决系数(coefficient of determination)也叫拟合优度,反映的是自变量x对因变量y的变动的解释的程度.越接近于1,说明模型拟合得越好.