openGauss AI特性创新实践课



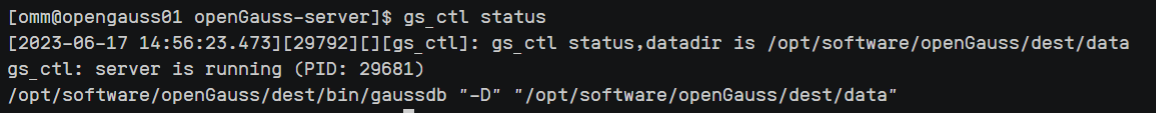
华为技术有限公司

# 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：

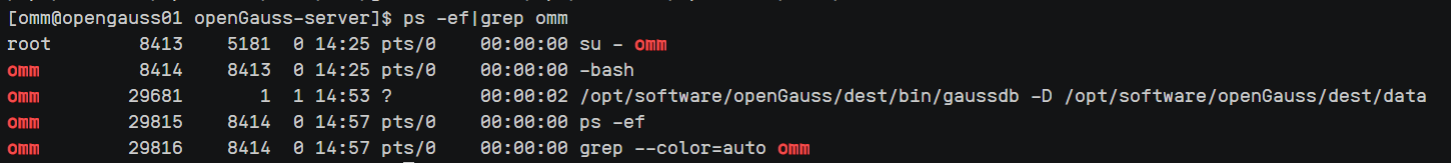
任务一：数据库状态验证

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？**

通过源码编译安装数据库有以下几个原因：

1. **定制化安装**：通过源码编译安装数据库，可以根据实际需求进行定制化安装。你可以选择需要的功能和插件，以及配置参数来满足特定的应用场景和性能需求。这种方式可以最大程度地发挥数据库的性能和功能优势。
2. **最新版本和功能支持**：源码编译安装可以确保你获得最新的数据库版本和功能支持。当数据库发布新版本时，源码通常会更早地提供下载，而二进制发行版需要一些时间才能跟上。此外，通过源码编译安装还可以选择性地启用或禁用特定的功能模块，以满足你的需求。
3. **平台兼容性**：源码编译安装可以确保数据库能够在特定的操作系统和硬件平台上运行。通过根据目标平台的特性和要求进行编译，可以获得更好的兼容性和性能。这对于一些非常定制化的环境或嵌入式系统来说尤其重要。
4. **调优和性能优化**：通过源码编译安装，你可以对数据库进行更深入的调优和性能优化。你可以根据硬件配置和应用负载的需求进行参数设置和编译选项的调整，以获得更好的性能和效率。

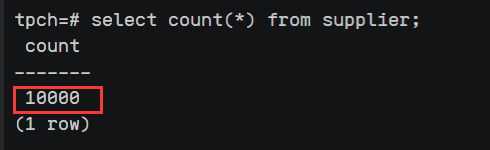
需要注意的是，源码编译安装相对于二进制发行版而言可能需要更多的时间和技术知识。它需要你具备一定的编译和配置能力，并且需要安装相关的开发工具和依赖库。因此，在选择源码编译安装时，需要权衡安装过程的复杂性和所带来的优势。

# 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：

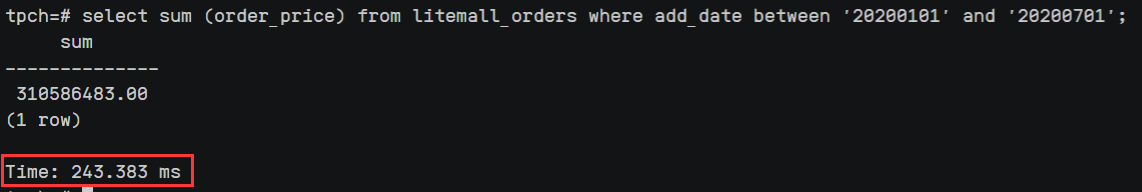
select count(\*) from supplier;;



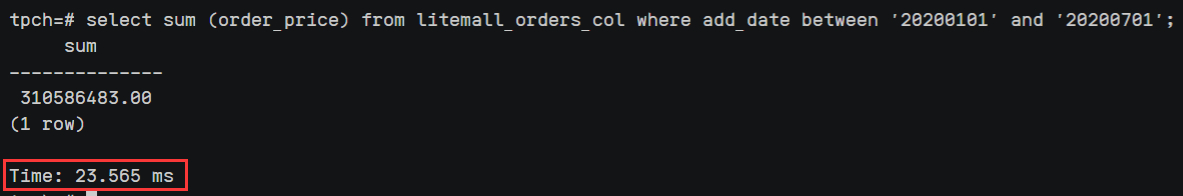
任务二：行存表与列存表执行效率对比

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';

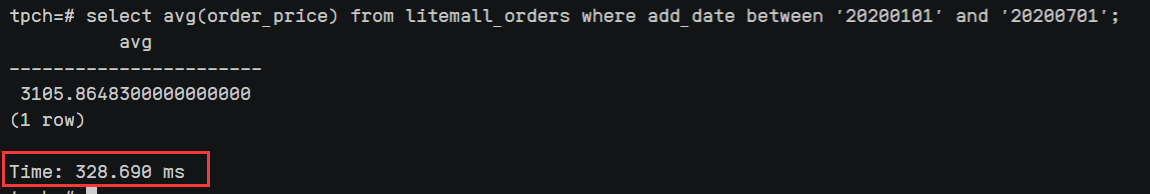


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

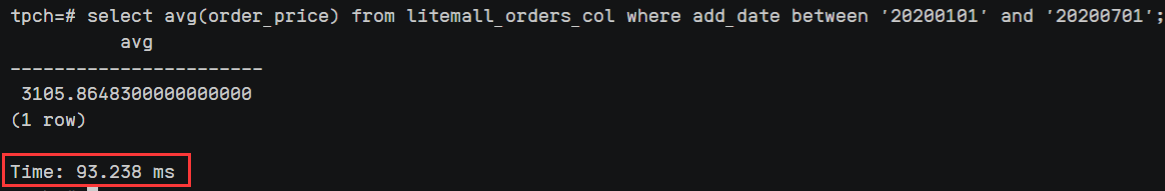


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

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

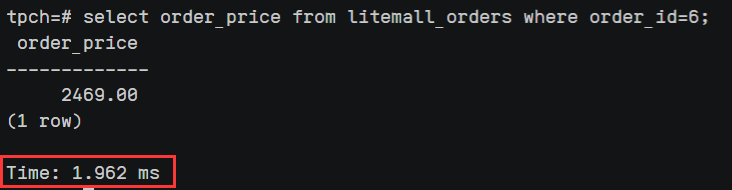


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

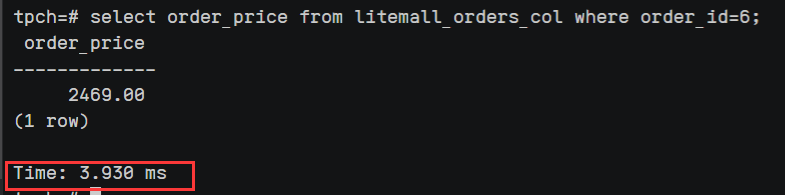


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

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

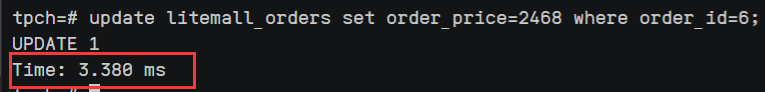


select order\_price from litemall\_orders\_col where order\_id=6;

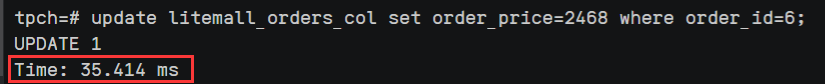


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

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

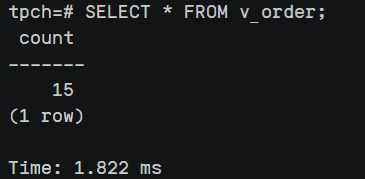
1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

SELECT \* FROM test\_view;



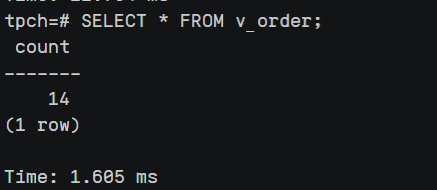
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



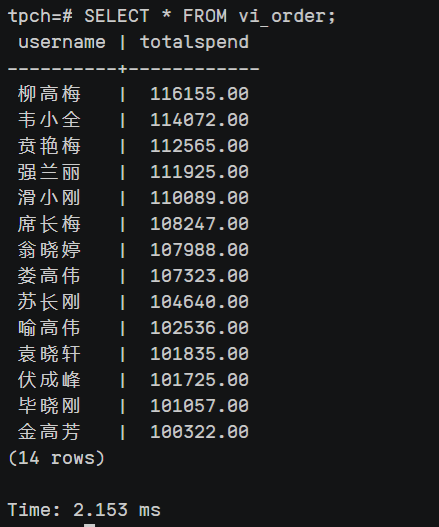
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

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（Row-based table）**和**列存表（Columnar table）**是两种不同的数据库存储方式，它们在执行相同的SQL语句时可能会有不同的执行时间。下面是一些常见情况：

* **行存表效率更高的情况：**

1. 单条记录读取：当需要查询或更新单个记录时，行存表的效率通常更高。因为行存表在物理上将相关字段存储在一起，可以更快地读取或更新整个记录。
2. 事务处理：对于需要频繁进行事务处理的场景，行存表通常更有效。行存表在处理事务时更加高效，因为它可以一次性读取或更新整个记录。
3. 多表关联：当需要进行多表关联查询时，行存表可能更具优势。行存表可以更快地获取整个记录的相关字段，减少了磁盘I/O和数据传输的开销。

* **列存表效率更高的情况：**

1. 聚合查询：当需要进行聚合操作，例如计算平均值、求和、统计等时，列存表通常更高效。因为列存表将同一列的数据存储在一起，可以只读取相关列，而不需要读取整个记录，从而减少了数据传输和处理的开销。
2. 大规模数据分析：对于大规模的数据分析和处理任务，列存表往往更加高效。列存表在处理大量数据时能够充分利用数据的压缩和向量化处理等优势，从而提高查询和分析的性能。
3. 大量列选择：当需要选择性地读取特定列的数据时，列存表通常更有效。列存表只需要读取所需的列，可以减少不必要的数据传输和磁盘I/O。

需要注意的是，行存表和列存表的效率受到多个因素的影响，包括数据量的大小、查询条件、索引的使用、硬件配置等。因此，在实际应用中，选择适合的存储方式需要考虑具体的场景和需求，以获得最佳的性能。

**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（Materialized View）**和**增量物化视图（Incremental Materialized View）**是两种不同类型的物化视图，它们有以下差别：

1. **数据更新方式：**

* 全量物化视图：全量物化视图在每次刷新时重新计算和获取完整的数据集，并将结果存储在物化视图中。这意味着全量物化视图需要重新计算和更新所有的数据，无论数据是否发生变化。
* 增量物化视图：增量物化视图只计算和存储数据的增量变化。在每次刷新时，增量物化视图会检测源表的变化，并只更新发生变化的数据。这样可以减少计算和存储的开销，提高刷新性能。

1. **存储和性能：**

* 全量物化视图：全量物化视图需要存储和维护完整的数据集，因此它们的存储需求更大。刷新时需要重新计算和加载所有数据，可能需要较长的时间。全量物化视图适用于数据变化频率较低或数据量较小的情况。
* 增量物化视图：增量物化视图只存储增量数据，因此它们的存储需求更低。刷新时只需要计算和加载发生变化的数据，因此刷新时间更短。增量物化视图适用于数据变化频率较高或数据量较大的情况。

1. **使用场景：**

* 全量物化视图：全量物化视图适用于静态数据或数据变化较少的情况。它们可以用于缓存复杂的查询结果，提高查询性能。全量物化视图通常用于离线或周期性地刷新数据。
* 增量物化视图：增量物化视图适用于数据变化频繁的情况。它们可以在刷新过程中只计算和更新变化的数据，减少计算和存储的开销。增量物化视图通常用于实时或近实时的数据刷新。

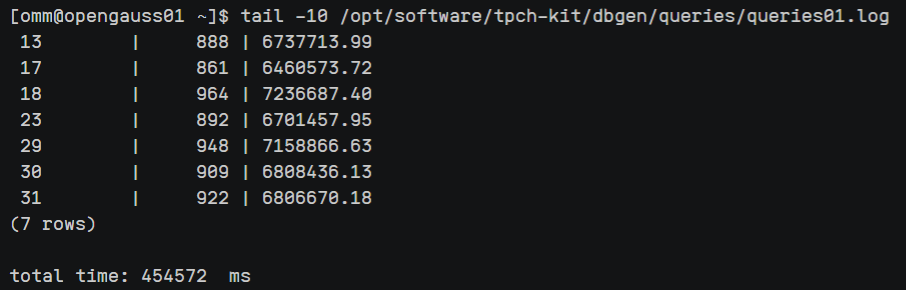
需要根据具体的业务需求和数据特点选择适合的物化视图类型。全量物化视图适用于较小规模的、静态的或数据变化较少的场景，而增量物化视图适用于较大规模的、动态的或数据变化频繁的场景。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

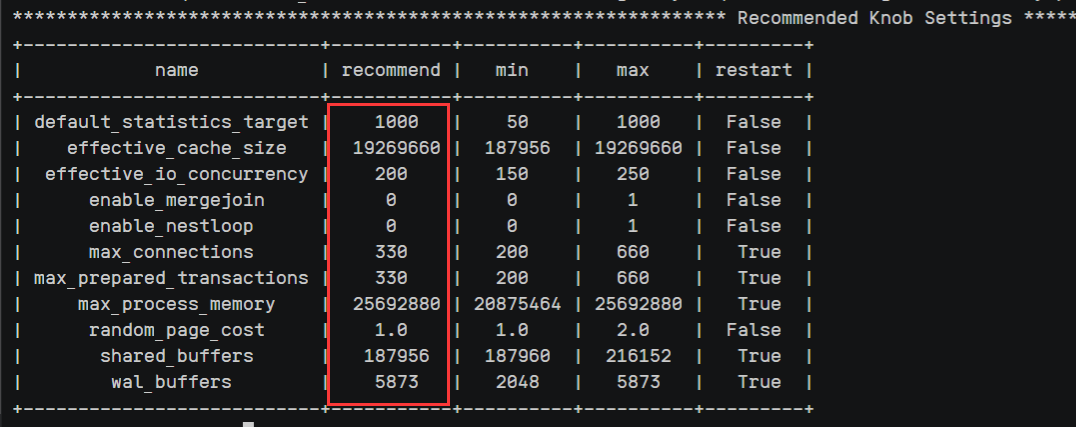
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

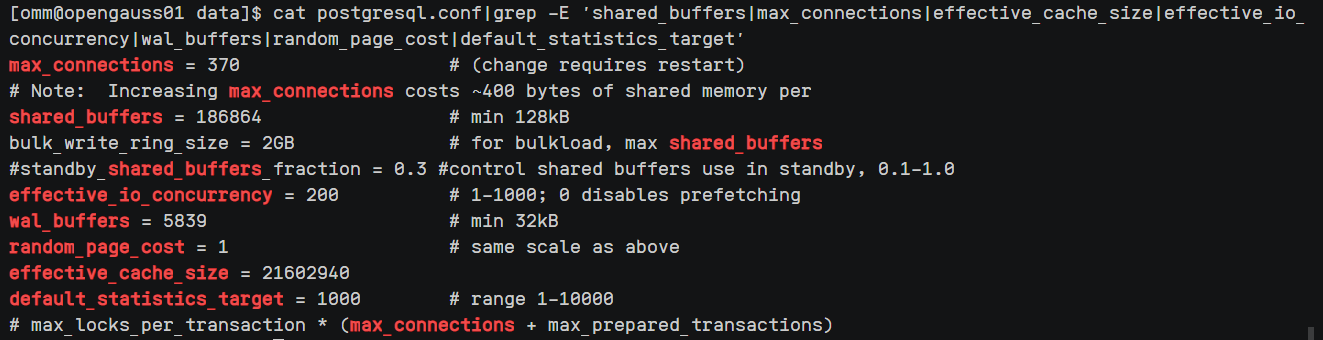
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

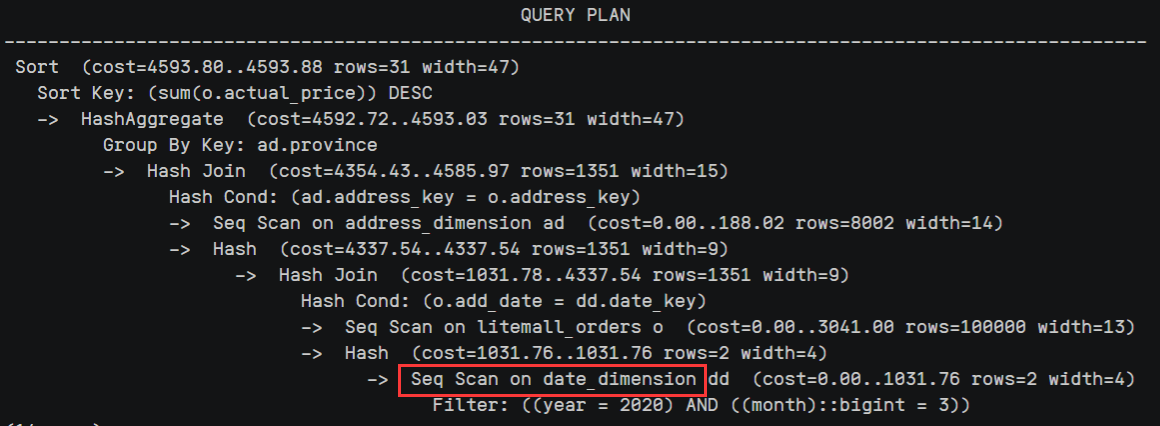
AND o.add\_date = dd.date\_key

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

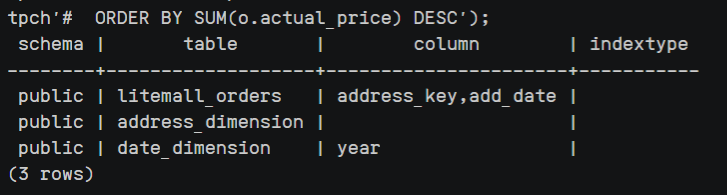
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

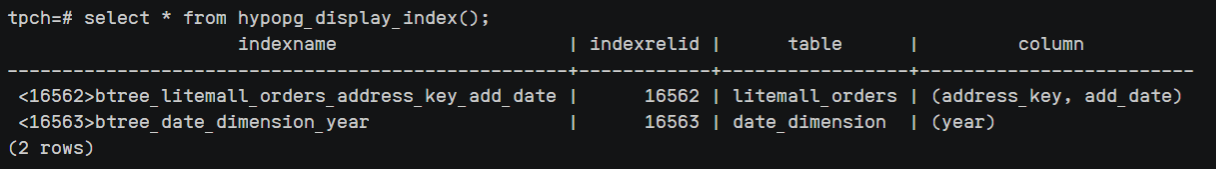
GROUP BY ad.province

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。

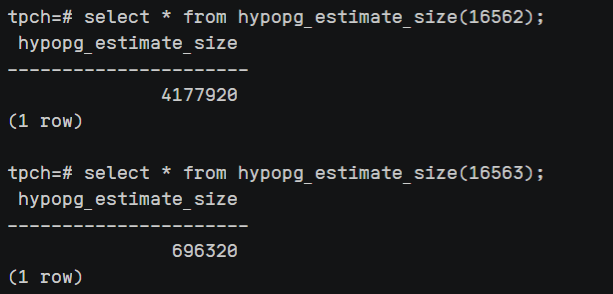
select \* from hypopg\_display\_index();



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

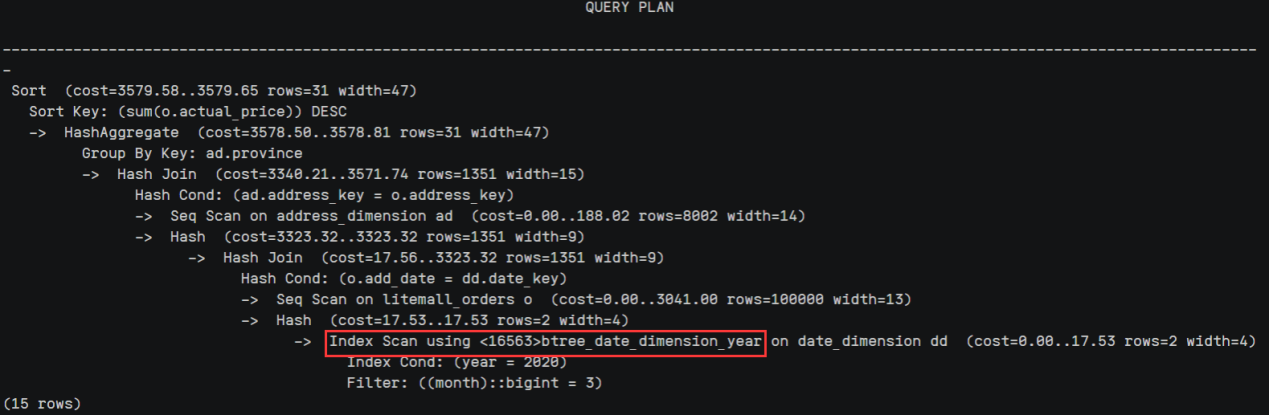
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

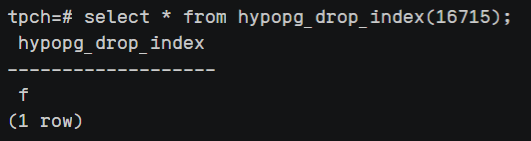
GROUP BY ad.province

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



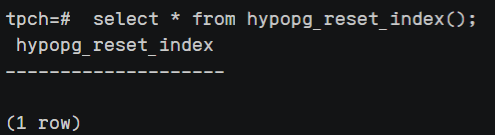
6. 删除某一个索引虚拟列，将执行结果截图。

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



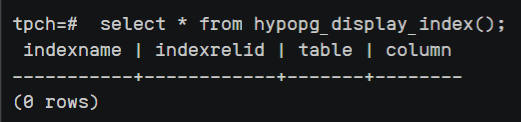
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

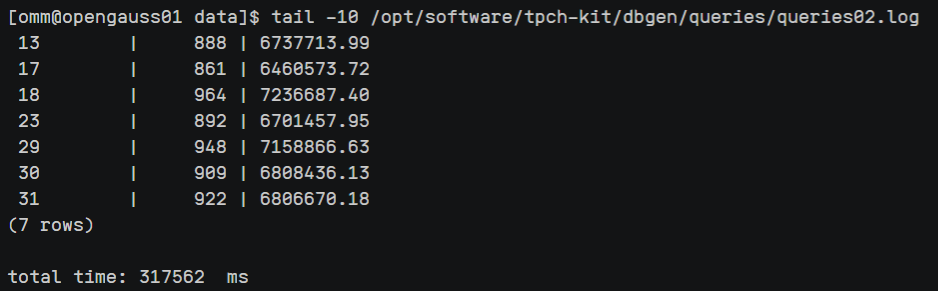
select \* from hypopg\_display\_index();



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：

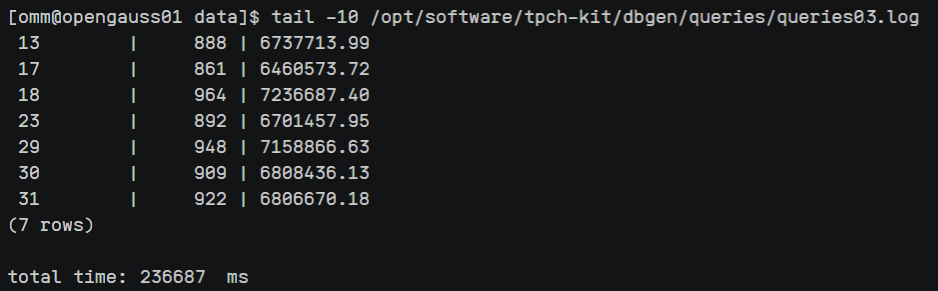
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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？**

根据X-Tuner给出的参数优化，对以下参数进行了优化：shared\_buffers、max\_connections、effective\_cache\_size、effective\_io\_concurrency、wal\_buffers、random\_page\_cost、default\_statistics\_target，对上述参数进行优化的原因如下：

1. **shared\_buffers**：这个参数决定了数据库系统中用于缓存数据和索引的内存大小。通过适当增大shared\_buffers，可以提高数据的内存缓存效果，减少磁盘I/O的次数，从而提高查询性能和响应速度。
2. **max\_connections**：该参数确定了数据库系统同时能够处理的最大客户端连接数。通过合理设置max\_connections，可以避免并发连接过多导致系统资源不足，影响系统的稳定性和性能。
3. **effective\_cache\_size**：这个参数用于指定系统可用的内存大小，供查询优化器估算磁盘I/O成本。通过准确设置effective\_cache\_size，可以帮助查询优化器更准确地选择合适的执行计划，提高查询性能。
4. **effective\_io\_concurrency**：该参数用于指定系统在进行并发磁盘I/O时的并发度。通过适当设置effective\_io\_concurrency，可以使系统更好地利用存储设备的并发读取能力，提高并发查询的性能。
5. **wal\_buffers**：该参数定义了用于写入WAL（Write-Ahead Log）的缓冲区大小。通过增大wal\_buffers，可以减少频繁的磁盘写入操作，提高事务提交的性能。
6. **random\_page\_cost**：这个参数用于指定随机读取相对于顺序读取的成本。通过适当调整random\_page\_cost，可以更准确地评估查询中的随机磁盘访问成本，从而选择更合适的执行计划。
7. **default\_statistics\_target**：该参数用于指定收集和使用统计信息的粒度。通过设置合理的default\_statistics\_target，可以提高查询优化器对表和索引的统计信息的准确性，从而生成更优化的查询执行计划。

优化这些参数可以根据数据库系统的实际硬件配置、负载情况和性能需求来提升数据库的整体性能和响应速度。每个参数的优化都有其具体的原因和优化目标，而X-Tuner通过自动化的方式，根据实时的性能指标和数据库的工作负载特点，推荐最佳的参数设置，以提高系统性能。

**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？**

使用索引可以为执行SQL语句带来以下好处：

1. 加快数据检索：索引能够提供快速的数据检索路径，减少了全表扫描的需求。通过使用适当的索引，可以快速定位到满足查询条件的数据，提高查询速度。
2. 减少磁盘I/O：索引存储了数据的副本，它们通常比完整的数据行小。当查询只需要读取索引列时，可以减少磁盘I/O的开销，提高查询性能。
3. 优化排序和分组：对于需要排序或分组的查询，索引可以提供已排序的数据，减少排序和分组操作的开销，提高执行速度。

除了使用索引和参数进行优化之外，还有以下方面可以对数据库进行优化：

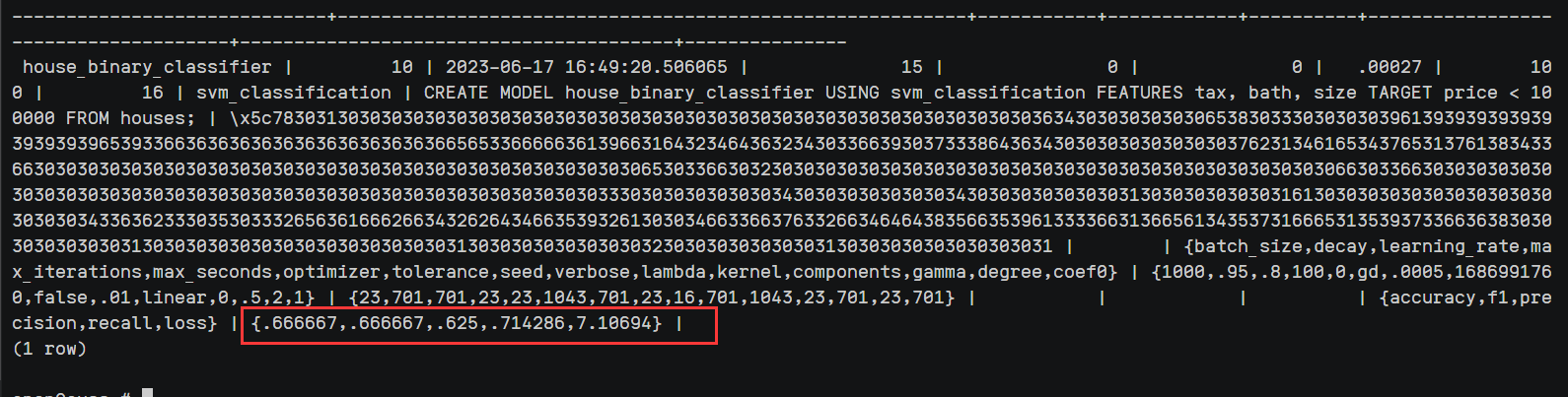
1. 查询优化：通过编写高效的查询语句、合理设计查询的逻辑和条件，可以减少不必要的计算和数据传输，提高查询性能。
2. 数据库设计优化：合理的数据库设计可以减少数据冗余、规范化数据模型，提高数据存储和查询的效率。
3. 缓存和缓冲区管理：通过合理配置缓存和缓冲区，可以减少磁盘I/O操作，提高数据的访问速度。
4. 并发控制和事务管理：合理的并发控制和事务管理可以提高数据库的并发性能和数据一致性。
5. 硬件和存储优化：选择适当的硬件设备和存储方案，如磁盘阵列、SSD等，可以提高数据库的读写性能。
6. 查询日志和性能监控：通过查询日志和性能监控工具，可以及时发现和解决潜在的性能问题，优化数据库的运行效率。
7. 定期维护和优化：定期进行数据库的维护和优化操作，如重建索引、优化查询计划、清理无用数据等，可以保持数据库的良好性能。

综合考虑这些方面进行数据库优化，可以提高数据库的性能、响应速度和可靠性，满足应用程序的需求。

# 关卡四、openGauss的DB4AI特性应用

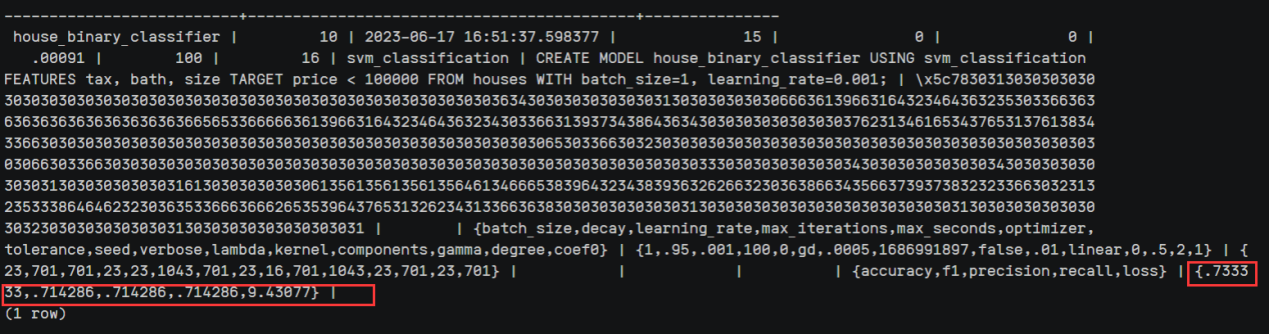
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



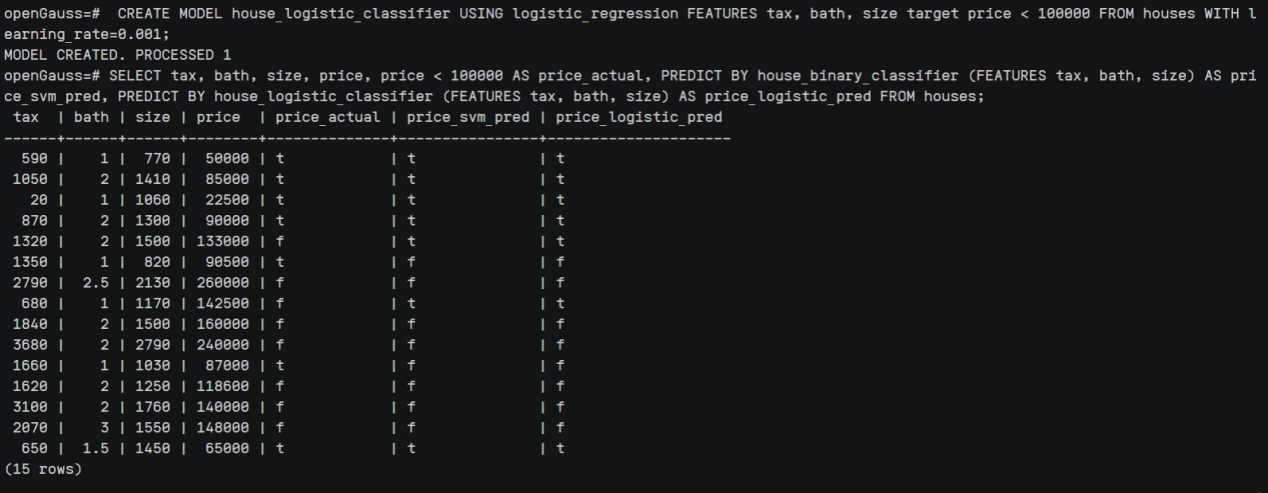
任务二：观察新模型的信息，将执行结果截图。

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？**

分类模型和回归模型是两种常见的机器学习模型，它们在目标和输出的类型上有所不同：

1. **目标类型：**

* 分类模型：分类模型用于预测离散类别或标签，将输入数据映射到事先定义好的类别中。例如，判断一封电子邮件是否是垃圾邮件、预测图像中的物体类别等。
* 回归模型：回归模型用于预测连续数值或数值范围，通过对输入数据建立一个数学函数，输出连续的数值。例如，预测房价、预测销售量等。

1. **输出类型：**

* 分类模型：分类模型的输出是离散的类别标签或概率分布，用于对新的输入数据进行分类或预测其所属类别的概率。常见的分类算法包括逻辑回归、决策树、支持向量机、朴素贝叶斯等。
* 回归模型：回归模型的输出是连续的数值，用于预测目标变量的具体数值。常见的回归算法包括线性回归、多项式回归、决策树回归、神经网络回归等。

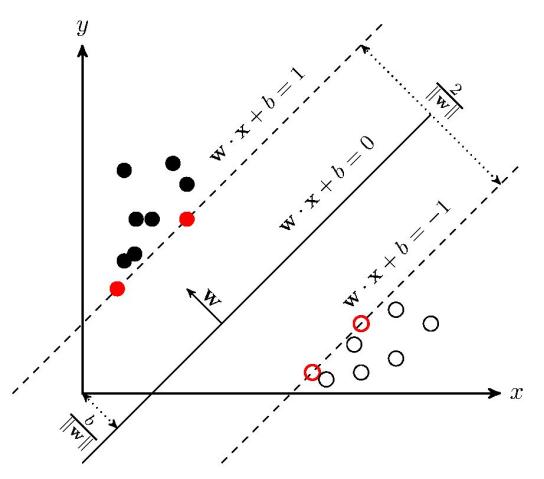
1. **模型评估：**

* 分类模型：分类模型的性能评估通常使用准确率、精确率、召回率、F1分数、ROC曲线等指标来衡量模型的分类性能。
* 回归模型：回归模型的性能评估通常使用均方误差（MSE）、均方根误差（RMSE）、平均绝对误差（MAE）、决定系数（R-squared）等指标来衡量模型的回归性能。

虽然分类模型和回归模型在目标和输出类型上有所不同，但它们都是基于训练数据进行模型学习，并用学习到的模型对新的输入数据进行预测或分类。在实际应用中，选择合适的模型类型取决于问题的性质和所需的预测结果。

**实践思考题2：什么是SVM算法？**

**支持向量机（Support Vector Machine）**是由Vapnik和他的合作者共同提出的一套学习算法，是统计学习理论的一种实现方法，它较好地实现了SRM原则。SVM通过引入**核函数**，将样本向量映射到高维特征空间，然后在高维空间中构造最优分类面，获得线性最优决策函数。



SVM可以通过控制**超平面的间隔度量**来抑制函数的过拟合；通过采用核函数巧妙解决了维数问题，避免了学习算法计算复杂度与样本维数的直接相关；也由于SRM原则的使用，SVM具有了良好的推广能力。

* **线性分类**

SVM定义**最优线性超平面**，并把寻找最优线性超平面转化为求解二次规划问题，进而基于Mercer定理，通过非线性映射，把样本空间映射到高维特征空间，从而使用线性方法解决样本空间中的高度非线性问题。支持向量机是针对二类别分类提出的。假设给定训练样本，存在分类超平面，为使分类面对所有样本正确分类且具备分类间隔，必须满足 **：**

|  |  |
| --- | --- |
|  | () |

可以计算出**分类间隔**为

|  |  |
| --- | --- |
|  | () |

要求最大分类间隔，即要求最小化。则求解最优分类超平面问题就可以表示成约束优化问题，即在式(2)的约束下，最小化函数

|  |  |
| --- | --- |
|  | () |

引入Lagrange函数：

|  |  |
| --- | --- |
|  | () |

其中，为 Lagrange 系数。将式(4)分别对 w 和 b 求偏导并令其等于 0，就可以将上述问题转化为简单的**对偶问题**。

|  |  |
| --- | --- |
|  | () |

将式(5)带入式(4)中，即可得到对偶最优化问题：求解下列函数的最大值

|  |  |
| --- | --- |
|  | () |

这是一个不等式约束下的二次函数极值问题(QP，Quadratic Programming)。根据Karush-Kuhn-Tucker(KKT)条件，该优化问题的解必须满足：

|  |  |
| --- | --- |
|  | () |

因此，多数样本对应的是为 0 的，把对应于使式(1)中等号成立的样本称为**支持向量（Support Vectors）**。在支持向量机算法中，支持向量是训练集中的关键元素，它们离决策边界最近。如果去掉其它所有的训练样本，再重新进行训练，将得到相同的分类面。

求解上述二次规划问题后，则**分类决策函数**可表示为

|  |  |
| --- | --- |
|  | () |

式中的求和只对支持向量进行，即只有不为零的对应的训练样本决定分类结果，而其它样本与分类结果无关。b\* 是分类阈值。

当训练样本集为**线性不可分**时，引入**非负松弛变量**，分类超平面的最优问题为

|  |  |
| --- | --- |
|  | () |

其对偶问题为对α求解下列函数的最大值：

|  |  |
| --- | --- |
|  | () |

其中 C>0 是一个常数，称为误差惩罚参数，它控制对错分样本惩罚的程度；是在训练样本线性不可分时引入的非负松弛变量。

当样本线性不可分时，分类决策函数也可表示为式(8)的形式。

* **非线性分类**

对于非线性分类问题，则采用适当的内积函数就可以实现某一非线性变换后的线性分类，本文中使用**高斯核函数**，定义如下：

|  |  |
| --- | --- |
|  | () |

式中为高斯核的带宽（width），此时优化的目标函数变为

|  |  |
| --- | --- |
|  | () |

而相应的分类决策函数表示为

|  |  |
| --- | --- |
|  | () |

以上的分类决策函数就是支持向量机。

我们可以看到，把原问题转化为对偶问题，使得计算的复杂度不再取决于空间维数，而是取决于样本数，尤其是**样本中的支持向量数**，支持向量机的这个特点使它能有效的对付高维问题。

**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？**

在分类问题中，常用的评价指标包括准确率、精确率、召回率、F1分数和ROC曲线等。下面对它们进行逐个说明：

1. **准确率（Accuracy）**：准确率是分类模型预测正确的样本数与总样本数之比，用于评估模型整体的分类准确程度。准确率计算公式为：准确率 = 预测正确的样本数 / 总样本数。准确率越高表示模型的分类性能越好，但在不均衡数据集中，准确率可能会受到样本分布的影响。
2. **精确率（Precision）**：精确率是指分类器预测为正类别的样本中，真正为正类别的样本所占的比例。精确率计算公式为：精确率 = 真正类别的样本数 / 预测为正类别的样本数。精确率衡量了模型在预测为正类别时的准确性，较高的精确率表示模型预测的正类别更可靠。
3. **召回率（Recall）**：召回率是指分类器能够正确预测为正类别的样本数与实际正类别样本数之比。召回率计算公式为：召回率 = 真正类别的样本数 / 实际正类别的样本数。召回率衡量了模型对于正类别样本的识别能力，较高的召回率表示模型能够更好地发现正类别样本。
4. **F1分数（F1 Score）**：F1分数是精确率和召回率的综合评价指标，综合考虑了模型的准确性和识别能力。F1分数是精确率和召回率的调和均值，计算公式为：F1分数 = 2 \* (精确率 \* 召回率) / (精确率 + 召回率)。F1分数的取值范围在0和1之间，越接近1表示模型的综合性能越好。
5. **ROC曲线（Receiver Operating Characteristic curve）**：ROC曲线是衡量分类模型性能的一种图形化指标。ROC曲线以真正类别率（True Positive Rate，召回率）为纵轴，以假正类别率（False Positive Rate）为横轴绘制曲线。ROC曲线可以通过计算曲线下面积（AUC，Area Under the Curve）来评估模型的性能，AUC值越大表示模型的分类能力越好。

这些评价指标在不同场景和数据集中有不同的重要性，根据具体情况选择合适的评价指标进行评估和比较。在以下情况下，常用的评价指标选择如下：

1. **数据不平衡情况**：当数据集中正负样本不平衡时，准确率可能无法准确反映模型的性能。在这种情况下，精确率、召回率和F1分数更能提供有关模型的详细信息。关注召回率更多的情况下，可将召回率作为主要评价指标；而在注重准确性的场景中，可以将精确率作为主要评价指标。
2. **类别间代价不平衡情况**：当不同类别的错误代价不同或具有不同的重要性时，可以考虑使用代价敏感的评价指标，如代价敏感的精确率和召回率。
3. **阈值选择**：在二分类问题中，模型输出的概率可以通过选择不同的阈值进行二分类。在这种情况下，可以使用ROC曲线和AUC值作为评价指标，帮助选择最佳的阈值，平衡模型的真正类别率和假正类别率。

需要注意的是，评价指标的选择应根据具体问题的需求和关注点进行权衡。在实际应用中，可以结合多个指标综合评估模型的性能，并根据问题的特点和应用场景做出合适的选择。

**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？**

在回归问题中，通常使用以下评价指标来评估模型的性能：

1. **均方误差（Mean Squared Error，MSE）**：MSE是最常见的回归问题评价指标之一。它计算了预测值与真实值之间差异的平方的平均值。MSE越小，表示模型的预测越接近真实值。然而，MSE的值可能受到异常值的影响。
2. **均方根误差（Root Mean Squared Error，RMSE）**：RMSE是MSE的平方根，它在度量预测误差时与原始数据具有相同的单位。RMSE同样是常用的评价指标，它能够衡量模型的预测误差的平均值。
3. **平均绝对误差（Mean Absolute Error，MAE）**：MAE计算预测值与真实值之间差异的绝对值的平均值。与MSE不同，MAE对异常值不敏感，因为它没有对差异进行平方运算。MAE越小，表示模型的预测越准确。
4. **决定系数（Coefficient of Determination，R-squared）**：决定系数是用来衡量模型对观测数据的拟合程度。它的取值范围从0到1，越接近1表示模型对数据的解释能力越强。决定系数可以解释因变量中方差的百分比，它是通过计算预测值和真实值之间的方差比例得出的。
5. **解释方差分数（Explained Variance Score）**：解释方差分数衡量了模型对数据方差的解释能力。它表示模型能够解释响应变量中方差的百分比。解释方差分数的取值范围也是从0到1，越接近1表示模型的解释能力越好。

这些评价指标在回归问题中经常被使用，可以根据具体情况选择适合的指标来评估模型的性能。