47丨如何利用SQL对零售数据进行分析?

我们通过 OLTP 系统实时捕捉到了用户的数据，还需要在 OLAP 系统中对它们进行分析。之前我们讲解了如何对数据进行清洗，以及如何对分散在不同地方的数据进行集成，今天我们来看下如何使用 SQL 分析这些数据。

关于这部分内容，今天我们一起来学习下：

1. 使用 SQL 进行数据分析都有哪几种姿势?
2. 如何通过关联规则挖掘零售数据中的频繁项集?
3. 如何使用 SQL+Python 完成零售数据的关联分析?

使用 SQL 进行数据分析的 5 种姿势

在 DBMS 中，有些数据库管理系统很好地集成了 BI 工具，可以方便我们对收集的数据进行商业分析。

SQL Server 提供了 BI 分析工具，我们可以通过使用 SQL Server 中的 Analysis Services 完成数据挖掘任务。SQL Server 内置了多种数据挖掘算法，比如常用的 EM、K-Means 聚类算法、决策树、朴素贝叶斯和逻辑回归等分类算法，以及神经网络等模型。我们还可以对这些算法模型进行可视化效果呈现，帮我们优化和评估算法模型的好坏。

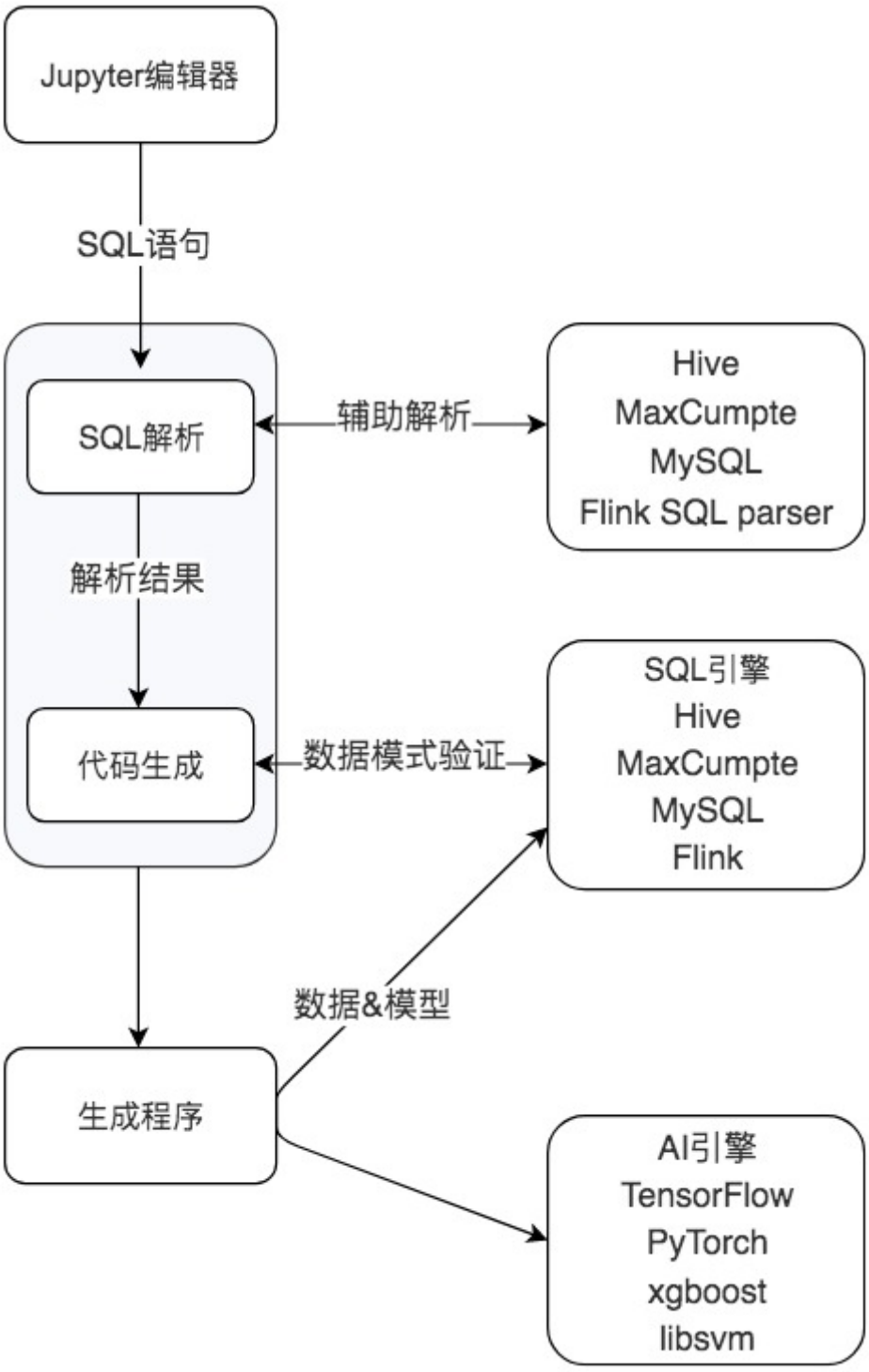
PostgreSQL 是免费开源的对象 - 关系数据库（ORDBMS），它的稳定性非常强，功能强大，在 OLTP 和 OLAP 系统上表现都非常出色。同时在机器学习上，配合 Madlib 项目可以让 PostgreSQL 如虎添翼。Madlib 包括了多种机器学习算法，比如分类、聚类、文本分析、回归分析、关联规则挖掘和验证分析等功能。这样我们可以通过使用 SQL，在 PostgreSQL 中使用各种机器学习算法模型，帮我们进行数据挖掘和分析。

2018 年 Google 将机器学习（Machine Learning）工具集成到了 BigQuery 中，发布了 BigQuery ML，这样开发者就可以在大型的结构化或半结构化的数据集上构建和使用机器学习模型。通过 BigQuery 控制台，开发者可以像使用 SQL 语句一样来完成机器学习模型的训练和预测。

SQLFlow 是蚂蚁金服于 2019 年开源的机器学习工具，我们通过使用 SQL 就可以完成机器学习算法的调用，你可以将 SQLFlow 理解为机器学习的翻译器。我们在 SELECT 之后加上

TRAIN 从句就可以完成机器学习模型的训练，在 SELECT 语句之后加上 PREDICT 就可以使用模型来进行预测。这些算法模型既包括了传统的机器学习模型，也包括了基于 Tensorflow、PyTorch 等框架的深度学习模型。

从下图中你也能看出 SQLFlow 的使用过程，首先我们可以通过 Jupyter notebook 来完成 SQL 语句的交互。SQLFlow 支持了多种 SQL 引擎，包括 MySQL、Oracle、Hive、SparkSQL 和 Flink 等，这样我们就可以通过 SQL 语句从这些 DBMS 中抽取数据，然后选择想要进行的机器学习算法（包括传统机器学习和深度学习模型）进行训练和预测。不过这个工具刚刚上线，工具、文档、社区还有很多需要完善的地方。



最后一个方法是 SQL+Python，也是我们今天要讲解的内容。刚才介绍的工具可以说既是 SQL 查询数据的入口，也是数据分析、机器学习的入口。不过这些模块耦合度高，也可能存在使用的问题。一方面工具会很大，比如在安装 SQLFlow 的时候，采用 Docker 方式（下图为使用 Docker 安装 sqlflow 的过程）进行安装，整体需要下载的文件会超过 2G。同时，在进行机器学习算法调参、优化的时候也存在灵活度差的情况。因此最直接的方式，还是将 SQL 与机器学习模块分开，采用 SQL 读取数据，然后通过 Python 来进行机器学习的处理。

```
Unable to find image 'sqlflow/sqlflow:latest' locally
latest: Pulling from sqlflow/sqlflow
16c48d79e9cc: Pull complete
3c654ad3ed7d: Pull complete
6276f4f9c29d: Pull complete
a4bd43ad48ce: Pull complete
d0d229c4ff68: Pull complete
742fb9c5b69f: Pull complete
7cfcaaf1157c: Pull complete
ba6f9f9bd917: Downloading  41.89MB/504.3MB
02bad18f1fe2: Download complete
dcc3ebbf16e3: Download complete
256920955058: Download complete
```

案例：挖掘零售数据中的频繁项集与关联规则

刚才我们讲解了如何通过 SQL 来完成数据分析（机器学习）的 5 种姿势，下面我们还需要通过一个案例来进行具体的讲解。

我们要分析的是购物篮问题，采用的技术为关联分析。它可以帮我们在大量的数据集中找到商品之间的关联关系，从而挖掘出经常被人们购买的商品组合，一个经典的例子就是“啤酒和尿布”的例子。

今天我们的数据集来自于一个面包店的 21293 笔订单，字段包括了 Date（日期）、Time（时间）、Transaction（交易 ID）以及 Item(商品名称)。其中交易 ID 的范围是 [1,9684]，在这中间也有一些交易 ID 是空缺的，同一笔交易中存在商品重复的情况。除此以外，有些交易是没有商品的，也就是对应的 Item 为 NONE。具体的数据集你可以从 GitHub上下载。

我们采用的关联分析算法是 Apriori 算法，它帮我们查找频繁项集，首先我们需要先明白什么是频繁项集。

频繁项集就是支持度大于等于最小支持度阈值的项集，小于这个最小值支持度的项目就是非频繁项集，而大于等于最小支持度的项集就是频繁项集。支持度是个百分比，指的是某个商品组合出现的次数与总次数之间的比例。支持度越高，代表这个组合出现的频率越大。

我们来看个例子理解一下，下面是 5 笔用户的订单，以及每笔订单购买的商品：

订单编号	购买的商品
1	牛奶、面包、尿布
2	可乐、面包、尿布、啤酒
3	牛奶、尿布、啤酒、鸡蛋
4	面包、牛奶、尿布、啤酒
5	面包、牛奶、尿布、可乐

在这个例子中，“牛奶”出现了 4 次，那么这 5 笔订单中“牛奶”的支持度就是 4/5=0.8。同样“牛奶 + 面包”出现了 3 次，那么这 5 笔订单中“牛奶 + 面包”的支持度就是 3/5=0.6。

同时，我们还需要理解一个概念叫做“置信度”，它表示的是当你购买了商品 A，会有多大的概率购买商品 B，在这个例子中，置信度（牛奶→啤酒）=2/4=0.5，代表如果你购买了牛奶，会有 50% 的概率会购买啤酒；置信度（啤酒→牛奶）=2/3=0.67，代表如果你购买了啤酒，有 67% 的概率会购买牛奶。

所以说置信度是个条件概念，指的是在 A 发生的情况下，B 发生的概率是多少。

我们在计算关联关系的时候，往往需要规定最小支持度和最小置信度，这样才可以寻找大于等于最小支持度的频繁项集，以及在频繁项集的基础上，大于等于最小置信度的关联规则。

使用 SQL+Python 完成零售数据的关联分析

针对上面的零售数据关联分析的案例，我们可以使用工具自带的关联规则进行分析，比如使用 SQL Server Analysis Services 的多维数据分析，或者是在 Madlib、BigQuery ML、SQLFlow 工具中都可以找到相应的关联规则，通过写 SQL 的方式就可以完成关联规则的调用。

除此以外，我们还可以直接使用 SQL 完成数据的查询，然后通过 Python 的机器学习工具包完成关联分析。下面我们通过之前讲解的 SQLAlchemy 来完成 SQL 查询，使用 efficient_apriori 工具包的 Apriori 算法。整个工程一共包括 3 个部分。

第一个部分为数据加载，首先我们通过 sql.create_engine 创建 SQL 连接，然后从 bread_basket 数据表中读取全部的数据加载到 data 中。这里需要配置你的 MySQL 账户名和密码

第二步为数据预处理，因为数据中存在无效的数据，比如 item 为 NONE 的情况，同时 Item 的大小写格式不统一，因此我们需要先将 Item 字段都转换为小写的形式，然后去掉 Item 字段中数值为 none 的项。在数据预处理中，我们还需要得到一个 transactions 数组，里面包括了每笔订单的信息，其中每笔订单是以集合的形式进行存储的，这样相同的订单中 item 就不存在重复的情况，同时也可以使用 Apriori 工具包直接进行计算。

最后一步，使用 Apriori 工具包进行关联分析，这里我们设定了参数 min_support=0.02，min_confidence=0.5，也就是最小支持度为 0.02，最小置信度为 0.5。根据条件找出 transactions 中的频繁项集 itemsets 和关联规则 rules。

具体的代码如下：

复制代码

```
from efficient_apriori import apriori
import sqlalchemy as sql
import pandas as pd
# 数据加载
engine = sql.create_engine('mysql+mysqlconnector://root:passwd@localhost/wucai')
query = 'SELECT * FROM bread_basket'
data = pd.read_sql_query(query, engine)
# 统一小写
data['Item'] = data['Item'].str.lower()
# 去掉 none 项
data = data.drop(data[data.Item == 'none'].index)

# 得到一维数组 orders_series，并且将 Transaction 作为 index, value 为 Item 取值
orders_series = data.set_index('Transaction')['Item']
```

```
# 将数据集进行格式转换
transactions = []
temp_index = 0
for i, v in orders_series.items():
    if i != temp_index:
        temp_set = set()
        temp_index = i
        temp_set.add(v)
        transactions.append(temp_set)
    else:
        temp_set.add(v)
# 挖掘频繁项集和频繁规则
itemsets, rules = apriori(transactions, min_support=0.02,  min_confidence=0.5)
print('频繁项集：', itemsets)
print('关联规则：', rules)
```

运行结果：

复制代码

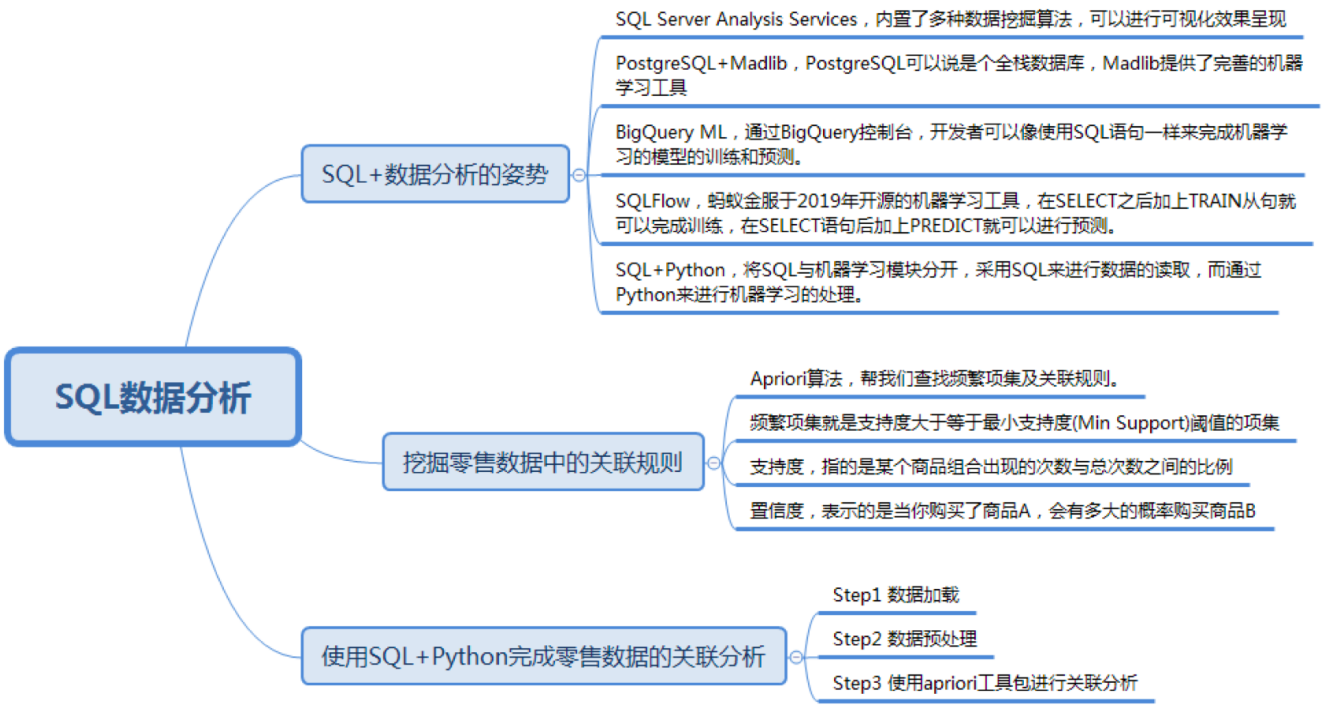
```
频繁项集：  {1: {('alfajores',): 344, ('bread',): 3096, ('brownie',): 379, ('cake',): 983
关联规则：  [{cake} -> {coffee}, {cookies} -> {coffee}, {hot chocolate} -> {coffee}, {jui
```

从结果中你能看到购物篮组合中，商品个数为 1 的频繁项集有 19 种，分别为面包、蛋糕、咖啡等。商品个数为 2 的频繁项集有 14 种，包括（面包，蛋糕），（面包，咖啡）等。其中关联规则有 8 种，包括了购买蛋糕的人也会购买咖啡，购买曲奇的同时也会购买咖啡等。

总结

通过 SQL 完成机器学习往往还是需要使用到 Python，因为数据分析是 Python 的擅长。通过今天的学习你应该能体会到采用 SQL 工具作为数据查询和分析的入口是一种数据全栈的思路，对于开发人员来说降低了数据分析的技术门槛。

如果你想要对机器学习或者数据分析算法有更深入的理解，也可以参考我的《数据分析实战 45 讲》专栏，相信在当今的数据时代，我们的业务增长会越来越依靠于 SQL 引擎 +AI 引擎。



我在文章中举了一个购物篮分析的例子，如下图所示，其中（牛奶、面包、尿布）的支持度是多少呢?

订单编号	购买的商品
1	牛奶、面包、尿布
2	可乐、面包、尿布、啤酒
3	牛奶、尿布、啤酒、鸡蛋
4	面包、牛奶、尿布、啤酒
5	面包、牛奶、尿布、可乐