南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

《计算机创新实践 II》

课程报告

题　　目： 工业大数据平台设计及关键技术研究

姓　　名： 吴季泫，陈孙兵，陈天乐

系　　别： 计算机系

专　　业： 计算机科学与技术

指导教师： 宋轩

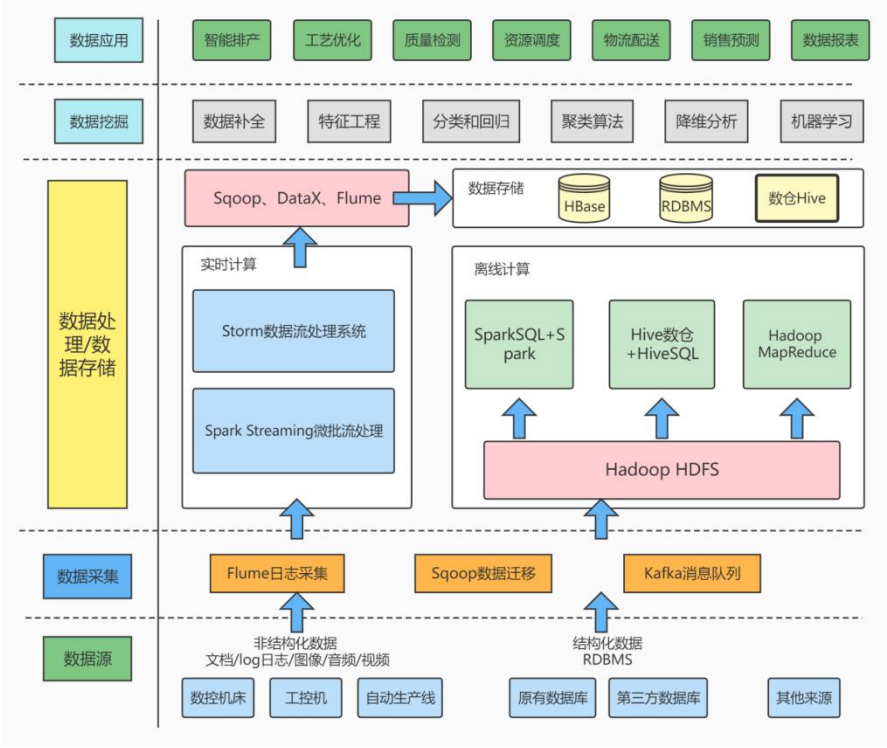
2023 年 6 月 13 日

目录

1.上学期回顾

1.1 了解什么是大数据平台以及一些相关组件和概念，例如 Hadoop、HDFS 分布式存储、Map Reduce 分布式计算等

1.2 设计大数据平台解决方案如下图所示



1.3 开发前后端可视化系统如下图所示

1.4 分析算法的初步构建，包括人工神经网络、线性回归、随机森林等常用的深度学习算法

1.5 服务器部署，包括后端系统的部署和大数据平台的搭建

本学期进展

第一次答辩 2023.3.30

2.1 数据离线迁移：

1、安装 ETL 工具 Sqoop

```
root@2a74f2e9b5c3:~# sqoop version
Warning: /usr/local/sqoop-1.4.7.bin__hadoop-2.6.0/../hbase does not exist! HBase imports will fail.
Please set $HBASE_HOME to the root of your HBase installation.
Warning: /usr/local/sqoop-1.4.7.bin__hadoop-2.6.0/../hcatalog does not exist! HCatalog jobs will fail.
Please set $HCAT_HOME to the root of your HCatalog installation.
Warning: /usr/local/sqoop-1.4.7.bin__hadoop-2.6.0/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
Warning: /usr/local/sqoop-1.4.7.bin__hadoop-2.6.0/../zookeeper does not exist! Accumulo imports will fail.
Please set $ZOOKEEPER_HOME to the root of your Zookeeper installation.
23/03/27 09:03:52 INFO sqoop.Sqoop: Running Sqoop version: 1.4.7
Sqoop 1.4.7
git commit id 2328971411f57f0cb683dfb79d19d4d19d185dd8
Compiled by maugli on Thu Dec 21 15:59:58 STD 2017
```

2、使用 sqoop 将数据从 MySQL 迁移到 Hive 上：

```
hive> show databases;
OK
default
Time taken: 9.375 seconds, Fetched: 1 row(s)
hive> show databases;
OK
default
Time taken: 0.032 seconds, Fetched: 1 row(s)
hive>
```

```
hive> select * from chip;
OK
1       TSSOP20 16      48      N/A     15      N/A     Cortex-M0       STM32F030F4P6   4       85      5       0       0       2.4-3.6
2       LQFP32  32      48      N/A     26      N/A     Cortex-M0       STM32F030K6T6   4       85      5       0       0       2.4-3.6
3       LQFP48  32      48      N/A     39      N/A     Cortex-M0       STM32F030C6T6   4       85      5       0       0       2.4-3.6
4       LQFP48  64      48      N/A     39      N/A     Cortex-M0       STM32F030C8T6   8       85      7       0       0       2.4-3.6
5       LQFP48  256     48      N/A     37      N/A     Cortex-M0       STM32F030CCT6   32      85      8       0       0       2.4-3.6
6       LQFP64  64      48      N/A     55      N/A     Cortex-M0       STM32F030R8T6   8       85      7       0       0       2.4-3.6
7       LQFP64  256     48      N/A     51      N/A     Cortex-M0       STM32F030RCT6   32      85      8       0       0       2.4-3.6
8       TSSOP20 32      48      N/A     15      N/A     Cortex-M0       STM32F070F6P6   6       85      5       0       1       2.4-3.6
9       LQFP48  32      48      N/A     37      N/A     Cortex-M0       STM32F070C6T6   6       85      5       0       1       2.4-3.6
10      LQFP48  128     48      N/A     37      N/A     Cortex-M0       STM32F070CBT6   16      85      8       0       1       2.4-3.6
11      LQFP64  128     48      N/A     51      N/A     Cortex-M0       STM32F070RBT6   16      85      8       0       1       2.4-3.6
12      TSSOP20 16      48      N/A     15      N/A     Cortex-M0       STM32F031F4P6   4       85      5       1       0       2-3.6
13      TSSOP20 32      48      N/A     15      N/A     Cortex-M0       STM32F031F6P6   4       85      5       1       0       2-3.6
14      WLCSP25 32      48      N/A     20      N/A     Cortex-M0       STM32F031E6Y6   4       85      5       1       0       2-3.6
15      UFQFPN28        16      48      N/A     23      N/A     Cortex-M0       STM32F031G4U6   4       85      5       1       0       2-3.6
16      UFQFPN28        32      48      N/A     23      N/A     Cortex-M0       STM32F031G6U6   4       85      5       1       0       2-3.6
17      UFQFPN32        16      48      N/A     27      N/A     Cortex-M0       STM32F031K4U6   4       85      5       1       0       2-3.6
18      UFQFPN32        32      48      N/A     27      N/A     Cortex-M0       STM32F031K6U6   4       85      5       1       0       2-3.6
```

2.2 算法构建

以下是有关大数据平台 AI 算法构建的报告，涉及决策树和人工神经网络等机器学习算法，并使用钢材生产相关的数据集进行了实验。

介绍

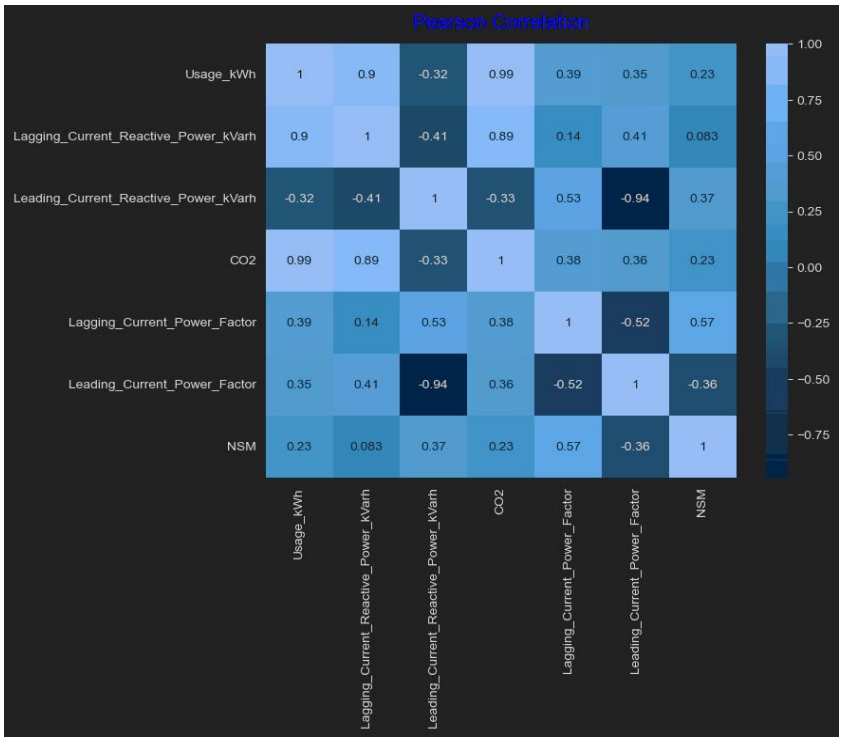
近年来，随着大数据时代的到来，数据量的爆炸性增长和处理难度的加大，对于数据的分析和利用提出了更高的要求。AI 算法在这一背景下得到了广泛的应用。钢材生产过程中需要对数据进行处理和分析，因此选择这个领域的数据集来实验，以验证算法的效果。

数据集

选取了一份钢材生产相关的数据集。该数据集包含了一些描述钢材制造过程的特征，例如能耗，碳排量等。我们将利用这些特征来预测未来生产中的能耗等信息。

	date	Usage_kWh	Lagging_Current_Reactive.Power_kVarh	Leading_Current_Reactive_Power_kVarh	CO2(tCO2)	Lagging_Current_Power_Factor	Leading_Current_Power_Factor	NSM
0	01/01/2018 00:15	3.17	2.95	0.0	0.0	73.21	100.0	900
1	01/01/2018 00:30	4.00	4.46	0.0	0.0	66.77	100.0	1800
2	01/01/2018 00:45	3.24	3.28	0.0	0.0	70.28	100.0	2700
3	01/01/2018 01:00	3.31	3.56	0.0	0.0	68.09	100.0	3600
4	01/01/2018 01:15	3.82	4.50	0.0	0.0	64.72	100.0	4500

数据可视化：



算法

决策树和人工神经网络等都是常见的机器学习算法，可以用于分类和回归问题，我们将借助这类经典机器学习算法来进行预测。

（ANN 运行示例）

```
history = model.fit(X_train, y_train, epochs=150)

# Get predictions on test set
y_pred = model.predict(X_test)

# Calculate R² score
r2 = r2_score(y_test, y_pred)

print('R² score:', r2)
```

```
Epoch 146/150
767/767 [==============================] - 1s 1ms/step - loss: 66.7353
Epoch 147/150
767/767 [==============================] - 1s 1ms/step - loss: 56.0606
Epoch 148/150
767/767 [==============================] - 1s 1ms/step - loss: 56.2434
Epoch 149/150
767/767 [==============================] - 1s 1ms/step - loss: 55.9813
Epoch 150/150
767/767 [==============================] - 1s 1ms/step - loss: 58.1066
329/329 [==============================] - 0s 924us/step
R² score: 0.962511354437561
```

实验

在实验中，我们使用 Python 中的 sklearn 和 keras 库实现了相关算法。首先，我们将数据集分为训练集和测试集，其中训练集占 80%，测试集占 20%。接下来，我们使用算法对数据集进行训练，然后使用测试集进行测试，并比较算法的效果。

algorithm	MAE	MSE
lr	2.551917608724714	17.769389051352 71
ridge	4.129658195653847	35.51065028271915
lasso	6.958401638582877	101.15479357255734
ANN	3.6620778787099795	42.07662790061998

2.3 连接大数据平台数据库

在 Python 中连接到大数据平台的数据库可以使用 PyHive 库。PyHive 是一个 Python 客户端库，用于在 Python 中连接到 Hadoop 和 Hive 数据库。在安装 PyHive 库之前，需要确保系统中安装了必要的依赖项，如 Python 和 Thrift。

PyHive 库提供了三种不同的客户端驱动程序：HiveServer 和 Impala。在本文中，我们将使用 HiveServer 客户端驱动程序。在 Python 中使用 PyHive 库连接到 HiveServer 客户端驱动程序：

```
from pyhive import hive
# 建立连接
conn = hive.Connection(host='localhost', port=10000, username='your_username',
```

```
password='your_password', database='default')
# 创建游标
cursor = conn.cursor()
# 执行 Hive 命令
cursor.execute('SELECT * FROM my_table')
# 获取查询结果
results = cursor.fetchall()
# 关闭连接
conn.close()
```

我们使用了 PyHive 库连接到 HiveServer 客户端驱动程序，并且执行了一个查询，最后获取了查询结果。

在 Hive 中创建数据库和表格

在使用 Hive 进行数据操作之前，需要先在 Hive 中创建数据库和表格。Hive 是一种基于 Hadoop 的数据仓库系统，可以将结构化的数据映射到 Hadoop 上，并提供 SQL 查询的能力。例如：

```
-- 创建数据库
CREATE DATABASE my_database;
-- 使用数据库
USE my_database;
-- 创建表格
CREATE TABLE my_table (
    id INT,
    name STRING
);
```

在 Python 中使用 Hive 进行数据操作[5][6]

在 Python 中使用 PyHive 库连接到 HiveServer 客户端驱动程序之后，可以使用 Hive 查询语言（HiveQL）在 Hive 中进行数据操作。以下是在 Python 中使用 Hive 进行数据操作的示例：

```
from pyhive import hive
# 建立连接
conn = hive.Connection(host='localhost', port=10000, username='your_username',
password='your_password', database='my_database')
# 创建游标
cursor = conn.cursor()
```

除了使用 Python 连接和操作大数据平台，我们还可以使用其他工具和技术。例如，Hadoop 是一个分布式存储和处理大数据的平台，它可以用于存储和处理大规模的结构化和非结构化数据。Hadoop 包括 Hadoop 分布式文件系统（HDFS）和 Hadoop MapReduce 编程模型。

另一个常用的大数据平台是 Apache Spark，它是一个快速的、分布式的计算引擎，可以处理大规模数据并支持多种编程语言。Spark 提供了一个交互式 Shell，可以使用 Python 或 Scala

编写代码进行数据分析和处理。

除了 Hadoop 和 Spark，还有许多其他的大数据平台和工具，如 Apache Cassandra、Apache HBase、Apache Kafka、Elasticsearch 等等。每种平台都有其独特的优势和用途，需要根据实际需求进行选择。

在操作大数据平台时，需要注意以下几点：
1．熟悉大数据平台的基本架构和组件，了解数据的存储和处理方式，以便选择合适的工具和技术。
2.学习和掌握相关编程语言和框架，如 Python、Java、Scala、Hive、Pig 等。
3．确保数据安全，采取必要的安全措施，如访问控制、数据加密等。
4．了解数据分析和处理的最佳实践，遵循数据规范和标准化的最佳实践，确保数据质量和一致性。
5．随时跟踪和监控数据平台的性能和健康状况，及时进行维护和优化。

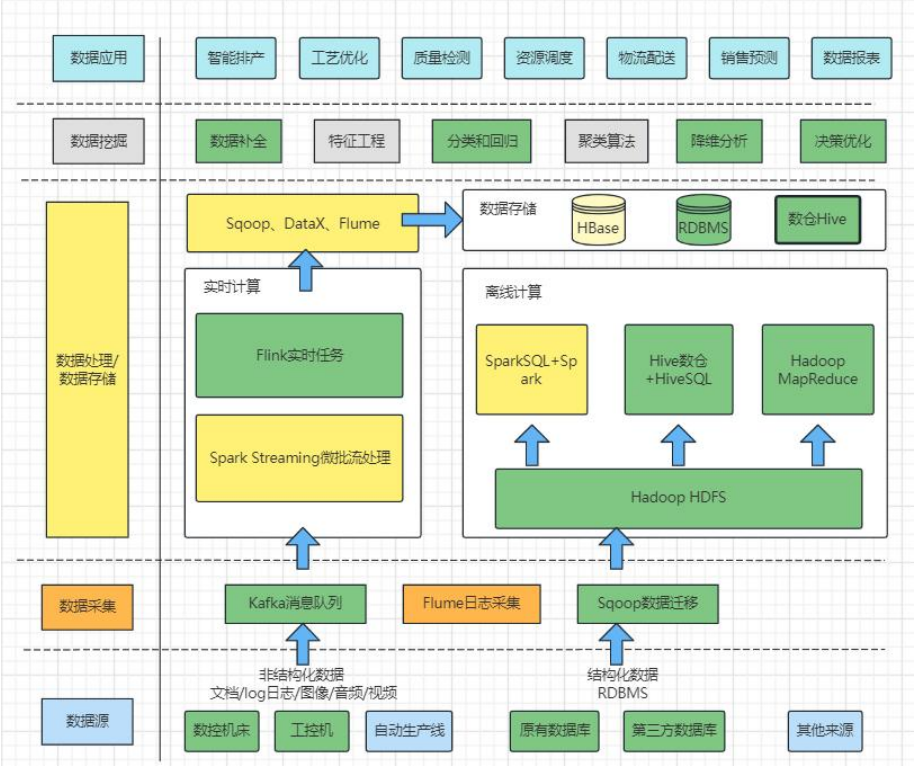
下一步计划

1、 完成 python 和平台的对接
2、 完成实时数据的传输
3、 完成部分分析算法的构建

3.第二次中期答辩 2023.5.2

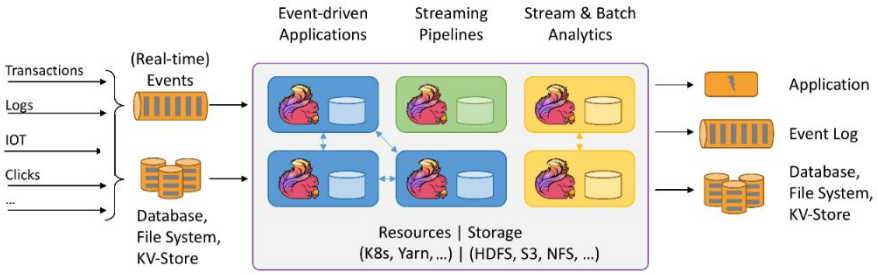
3.1 搭建大数据平台：

更新平台框架图如下，其中绿色的方框为当前已完成任务

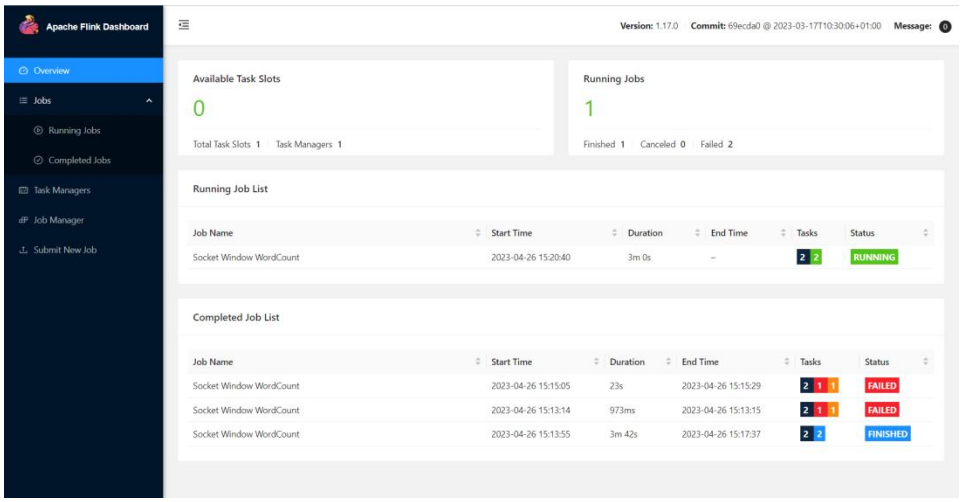


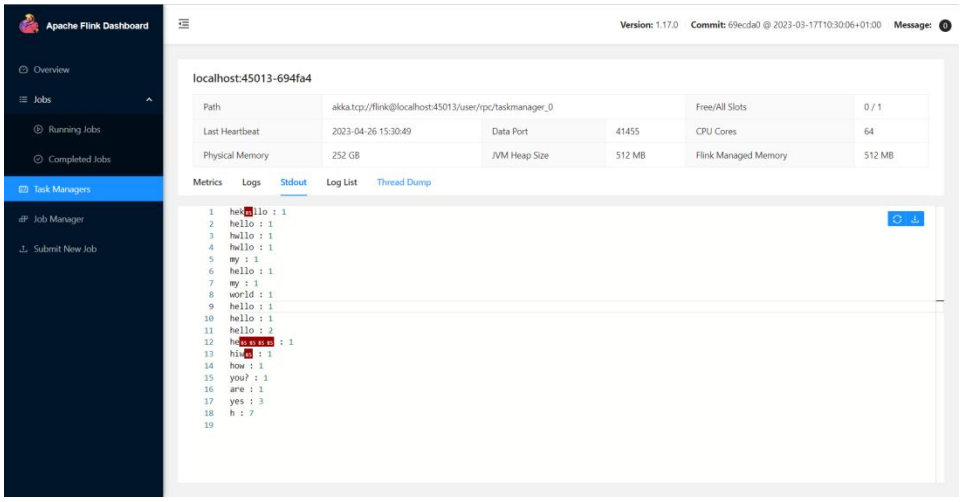
Flink

Flink 框架图（图源 Apache Flink 项目官网）



运行自带的 worldCount 程序，持续监听一个端口的数据，然后在另一个窗口执行 nc –l –p 9001 命令，持续向这个端口输出数据。通过然后通过 web UI 监控任务运行情况，可以看到当前运行任务的情况，以及对应的词频统计输出





然后通过 Java 构建自己的词频统计程序代码，通过获取 Flink 执行环境，并执行简单的词频统计逻辑，熟悉一些 API 接口和封装好的方法，在后续确定任务目标后可以更快的编写目标任务的处理逻辑代码

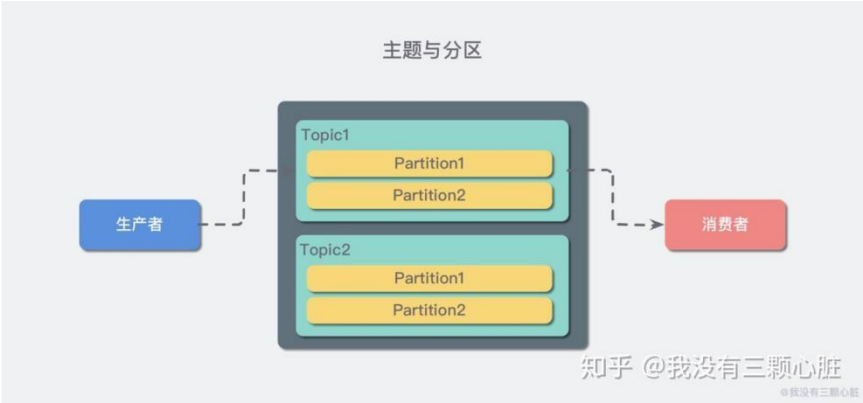
```
// 1. 创建流式执行环境
StreamExecutionEnvironment env =
        StreamExecutionEnvironment.getExecutionEnvironment();
// 2. 读取文本流
DataStreamSource<String> lineDSS = env.socketTextStream( hostname: "desktop",
        port: 9802);
// 3. 转换数据格式
SingleOutputStreamOperator<Tuple2<String, Long>> wordAndOne = lineDSS
        .flatMap((String line, Collector<String> words) -> {
            Arrays.stream(line.split( regex: " ")).forEach(words::collect);
        }) SingleOutputStreamOperator<String>
        .returns(Types.STRING)
        .map(word -> Tuple2.of(word, 1L)) SingleOutputStreamOperator<Tuple2<String, Long>>
        .returns(Types.TUPLE(Types.STRING, Types.LONG));
// 4. 分组
KeyedStream<Tuple2<String, Long>, String> wordAndOneKS = wordAndOne
        .keyBy(t -> t.f0);
// 5. 求和
SingleOutputStreamOperator<Tuple2<String, Long>> result = wordAndOneKS
        .sum( positionToSum: 1);
// 6. 打印
result.print();
// 7. 执行
env.execute();
```

Kafka 单机单群组

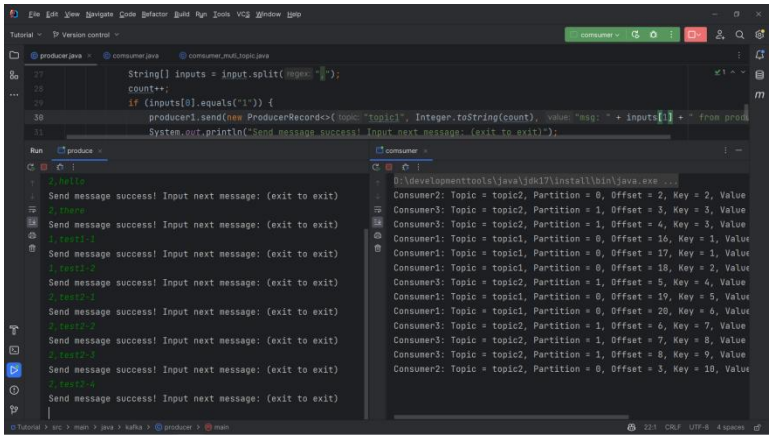
Kafka 相关概念：

- Producer：消息生产者，向 Broker 发送消息的客户端；
- Consumer：消息消费者，从 Broker 读取消息的客户端；
- Topic：Kafka 根据 topic 对消息进行归类，发布到 Kafka 集群的每条消息都需要指定一个 topic；
- Partition：物理上的概念，一个 topic 可以分为多个 partition，每个 partition 内部是有序的。在创建 topic 的时候可以指定创建多少个 topic，在 topic 创建之后也可以添加 partition，以提高服务器的负载能力。

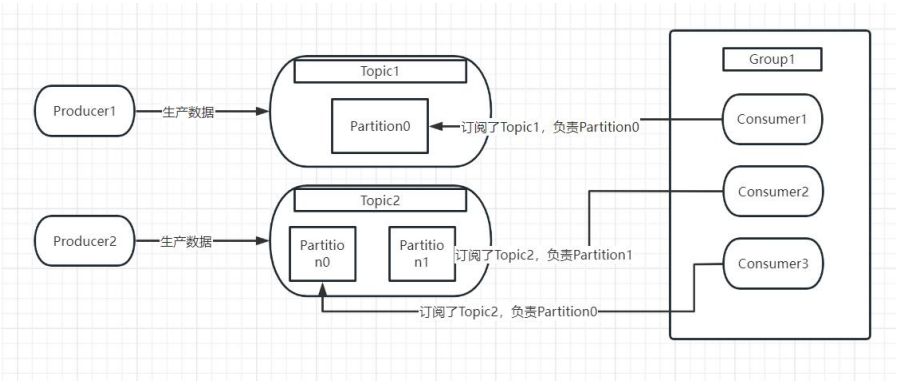
单机环境下，多个生产者和同一群组下多个消费者情境下：（图源网络）
生产者在产生数据时，可以指定向哪一个 topic 里面发送数据。消费者在创建之后可以选择订阅哪一个 topic。需要注意的是，订阅一个 topic 的消费者数量需要小于等于该 topic 的 partition 数量，否则就会导致有一些消费者是多余的，一直接不到消息而处于空闲状态。在完成订阅之后，消费者可以消费 topic 内的数据。



通过 Java 连接 Kafka 服务端口后执行生产者和消费者程序结果如下。



对订阅关系可以用图表表示如下：



Kafka 单机多群组

这里引入一个新的概念：ConsumerGroup。

- ConsumerGroup：每个 Consumer 属于一个特定的 Consumer Group，一条消息可以发送到多个不同的 Consumer Group，但是一个 Consumer Group 中只能有一个 Consumer 能够消费该消息；

单机环境下，多个生产者和不同群组下多个消费者情境下：（图源网络）

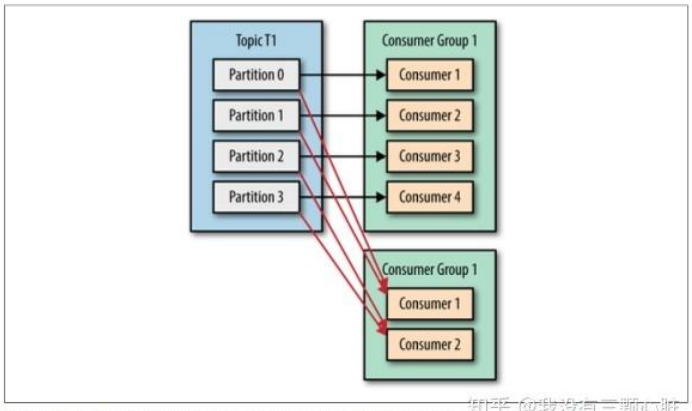
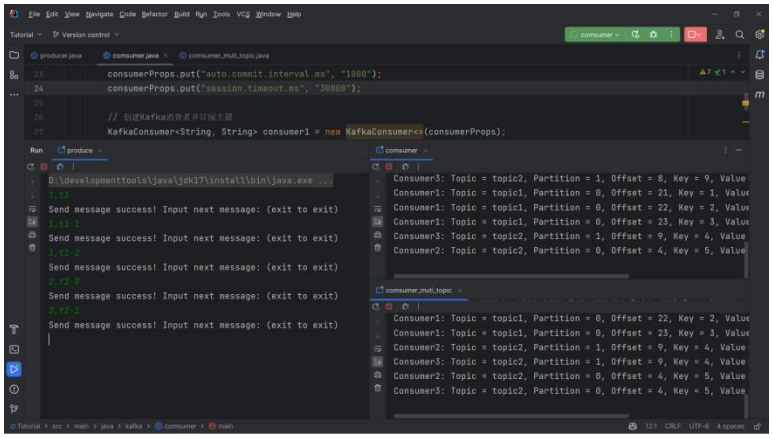


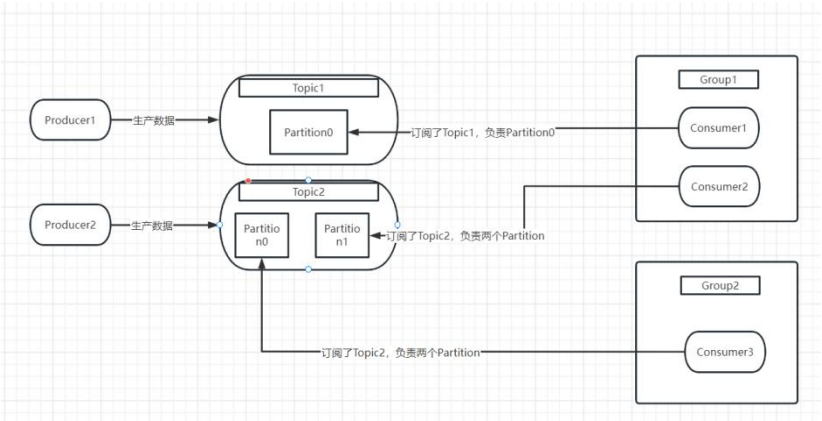
Figure 4-5. Adding a new consumer group ensures no messages are missed

通过 Java 连接 Kafka 服务端口后执行生产者和不同群组下的消费者程序结果如下，可以看到与之前同一群组下的结果是不一样的。具体体现在在

1. 同一群组下，Consumer2 和 Consumer3 虽然订阅了同一个 topic，但是它们负责的 Partition 不同，它们只能各种消费自己负责的 Partition 下的数据，因此对于一个新的发布到 Topic2 的数据，Kafka 会根据默认策略选择存放到哪一个 Partition 下，因此当新消息没有存放到它们负责的 Partition 时，它们是无法对新数据进行消费的。在这种场景下，对于 topic 的数据，它们是只能消费一部分的，不过这样也对应了一些特定的使用场景。
2. 在不同群组下，Consumer2 和 Consumer3 都订阅了同一个 topic，但是由于它们属于不同的群组，因此它们都必须负责这个 topic 下的所有 Partition，因此当有新的数据发布到 Topic2 的时候，它们都能消费最新到达的数据。



这个对应的订阅关系可以表示如下：



Hadoop 下的多源异构数据存储

在开始项目之前的思考：相较于 Hadoop 和本地文件系统储存数据，Hadoop 的优势是什么，为什么需要用 Hadoop 来进行文件的存储？

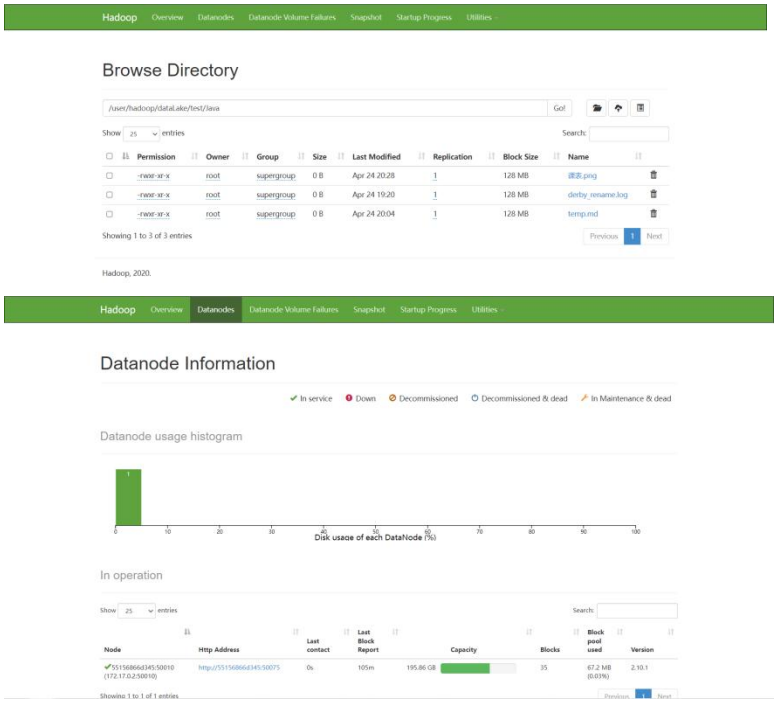
在进行调研之后，得出结论：主要原因是 Hadoop 是通过分布式来储存文件，这样做的好处是可以利用分布式来环境文件读写硬盘的 IO 瓶颈。举一个例子：假设我们有一份大文件需要处理，它的大小为 10GB。如果我们使用一般的 springboot 静态资源储存，我们需要将这个文件上传到服务器的硬盘上，然后再进行处理。而在处理过程中，我们需要从硬盘读取文件的数据，进行计算，然后将结果写回硬盘。这个过程中，读写硬盘的速度可能会成为瓶颈，影响整个处理过程的速度。而如果我们使用 Hadoop，我们可以将这个文件存储在 HDFS 上。在进行计算时，Hadoop 会将数据分块，每个节点上处理一部分数据，这样可以充分利用集群中的计算资源，提高计算速度。

其次原因是由于 HDFS 是分布式存储的，数据会被复制多份，保证数据的可靠性和容错性。在读写数据时，Hadoop 会自动选择最近的节点进行读写操作，避免了网络传输带来的延迟。因此，相比一般的 springboot 静态资源储存，Hadoop 具有更好的扩展性和高性能，特别是在处理大数据时能够显著提高数据处理速度。

在 Java 环境下，通过导入 Hadoop 相关 Maven 依赖，可以通过 Hadoop 相关 API 接口和方法对 Hadoop HDFS 文件系统进行操作。

```java
Main.java
58      // 测试从HDFS下载文件到本地
        no usages
59      @Test
60      public void testDownFileToLocal() throws IOException {...}
71
72      // 测试重命名文件
        no usages
73      @Test
74      public void testRenameFile() throws IOException {...}
85
86      // 测试删除文件
        no usages
87      @Test
88      public void testDelFile() throws IOException {...}
97
98      // 测试修改文件权限
        no usages
99      @Test
100     public void testModifyPermission() throws IOException {...}
113
114     // 测试递归查询文件夹下所有文件的信息
        no usages
115     @Test
116     public void testCheckFile() throws IOException {...}
```

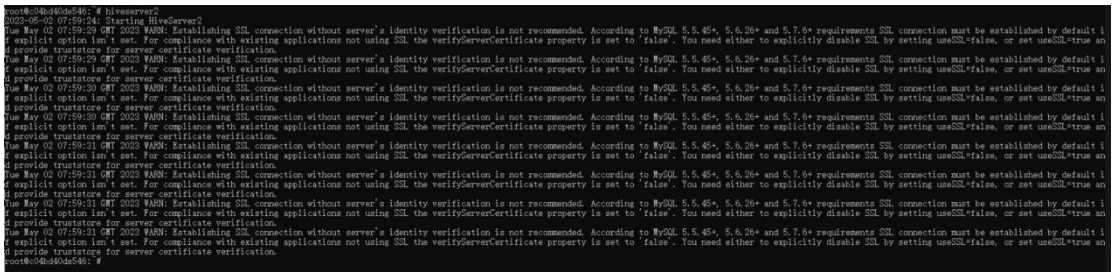
同时通过 Hadoop 的 Web UI 界面可以访问到 Hadoop 的 HDFS 文件目录结构，也可以查看当前运行节点的状态

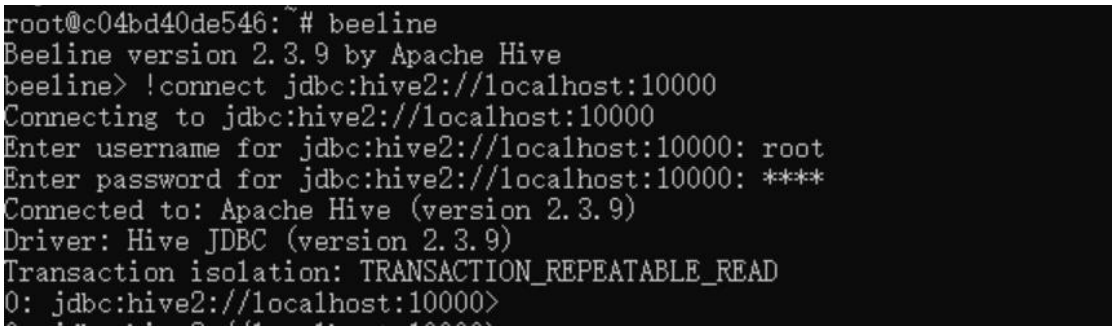


3.2 Python 连接大数据平台数据库

客户端连接 Hive 需要使用 HiveServer2。HiveServer2 是 HiveServer 的重写版本，HiveServer 不支持多个客户端的并发请求。当前 HiveServer2 是基于 Thrift RPC 实现的。它被设计用于为像 JDBC、ODBC 这样的开发 API 客户端提供更好的支持。Hive 0.11 版本引入的 HiveServer2。

首先启动 hiveserver2 并且用 beeline 工具在服务器内部测试开放的端口的可行性：





成功之后即可以利用 python 中的 pyhive 库去连接服务器映射的 docker 端口：

```python
from impala.dbapi import connect

conn = connect(host='172.18.36.63', port=1145, database='default', auth_mechanism='PLAIN')

cur = conn.cursor()

cur.execute('SHOW DATABASES')
print(cur.fetchall())

cur.execute('SHOW Tables')
print(cur.fetchall())
```

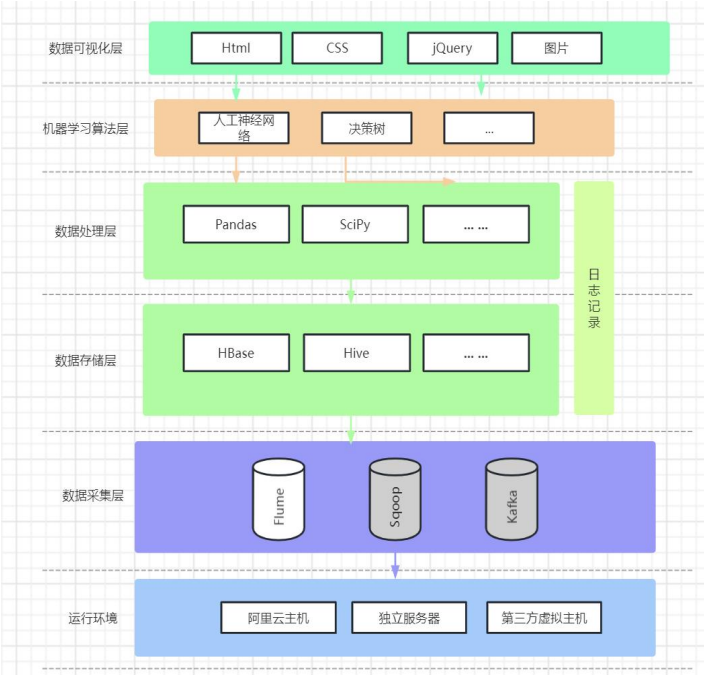
即可成功连接，成功查询到数据库：

```
/Users/apple/opt/anaconda3/envs/python3.7/bin/python /Users/apple/Desktop/大三下/创新实践/test.py
[('default',), ('mysqltest',), ('testme',)]
[]

Process finished with exit code 0
```

3.3 算法构建

设计的算法框架图如下：



回归算法

背景和目的

旨在分析使用不同的 AI 算法对钢铁制造工业中的能耗和碳排放量等信息进行预测的结果。本研究的目的是利用 AI 算法帮助工业企业降低能源消耗和减少碳排放，从而提高其可持续发展的能力。

数据集

数据集刻画了钢铁制造工业中的能耗以及碳排放量等信息。

	date	Usage_kWh	Lagging_Current_Reactive.Power_kVarh	Leading_Current_Reactive_Power_kVarh	CO2(tCO2)	Lagging_Current_Power_Factor	Leading_Current_Power_Factor	NSM
0	01/01/2018 00:15	3.17	2.95	0.0	0.0	73.21	100.0	900
1	01/01/2018 00:30	4.00	4.46	0.0	0.0	66.77	100.0	1800
2	01/01/2018 00:45	3.24	3.28	0.0	0.0	70.28	100.0	2700
3	01/01/2018 01:00	3.31	3.56	0.0	0.0	68.09	100.0	3600
4	01/01/2018 01:15	3.82	4.50	0.0	0.0	64.72	100.0	4500

模型选择和参数设置

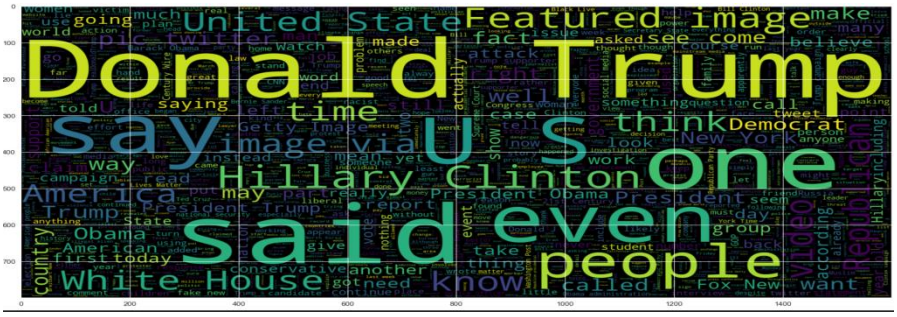
为了得到最优的预测结果，我们使用了四种不同的 AI 算法：线性回归（Linear Regression，LR）、岭回归（Ridge Regression，Ridge）、套索回归（Lasso Regression，Lasso）和人工神经网络（Artificial Neural Network，ANN）。对于每种算法，我们根据交叉验证的结果，调整了其超参数，如正则化系数、学习率、隐藏层数等。最终，我们选择了每种算法的最优参数进行训练和测试。

algorithm	MAE	MSE
lr	2.551917608724714	17.76938905135271
ridge	4.129658195653847	35.51065028271915
lasso	6.958401638582877	101.15479357255734
ANN	3.6620778787099795	42.07662790061998

分类算法

背景和目的
使用 LSTM 算法对新闻文章进行分类的结果。具体而言，我们使用一个包含标题、文本和发布时间的数据集，将文章分为真实和虚假两类。

数据集
数据集包含了一系列新闻文章，每篇文章都有标题、文本和发布时间等信息，并被标记为真实或虚假。



模型选择和参数设置
为了得到最优的分类结果，我们使用了 LSTM 算法。LSTM 是一种能够处理序列数据的深度神经网络模型，能够对输入数据的长期依赖关系进行建模。对于每个文章，我们将其标题和文本转化成词向量表示，并将其输入到 LSTM 模型中进行训练。我们根据交叉验证的结果，调整了 LSTM 模型的超参数，如 LSTM 层数、LSTM 输出维度、Dropout 率等。最终，我们选择了 LSTM 最优的参数进行训练和测试。

聚类算法

背景和目的
本报告旨在分析使用 K-means 算法对天气数据进行聚类的结果。具体而言，我们使用了来自 HPWREN 的天气数据集，包括时间戳、气压、温度等变量，旨在对这些数据进行聚类，以便更好地了解数据中存在的天气模式和特征。

数据集
数据集包含了来自 HPWREN 的天气数据，包括时间戳、气压、温度等变量。数据集包含 N 个样本，其中每个样本表示一个时间点的天气情况。为了进行聚类，我们选择了气压和温度

作为聚类的特征，同时将时间戳信息转化为数值型特征。

rowID	hpwren_timestamp	air_pressure	air_temp	avg_wind_direction	avg_wind_speed
0	2011-09-10 00:00:49	912.3	64.76	97.0	1.2
1	2011-09-10 00:01:49	912.3	63.86	161.0	0.8
2	2011-09-10 00:02:49	912.3	64.22	77.0	0.7
3	2011-09-10 00:03:49	912.3	64.4	89.0	1.2
4	2011-09-10 00:04:49	912.3	64.4	185.0	0.4
5	2011-09-10 00:05:49	912.3	63.5	76.0	2.5
6	2011-09-10 00:06:49	912.3	62.78	79.0	2.4
7	2011-09-10 00:07:49	912.3	62.42	86.0	2.0
8	2011-09-10 00:08:49	912.3	62.24	105.0	1.4

调度算法

背景和目的

本报告旨在分析使用 A*算法对制作圣诞卡的最佳方式进行调度的结果。具体而言，我们需要确定机械臂移动和打印颜色变化的最佳路径，以便在最短时间内完成圣诞卡的制作。本研究的目的是利用 AI 算法帮助用户优化制作流程，提高效率和质量。

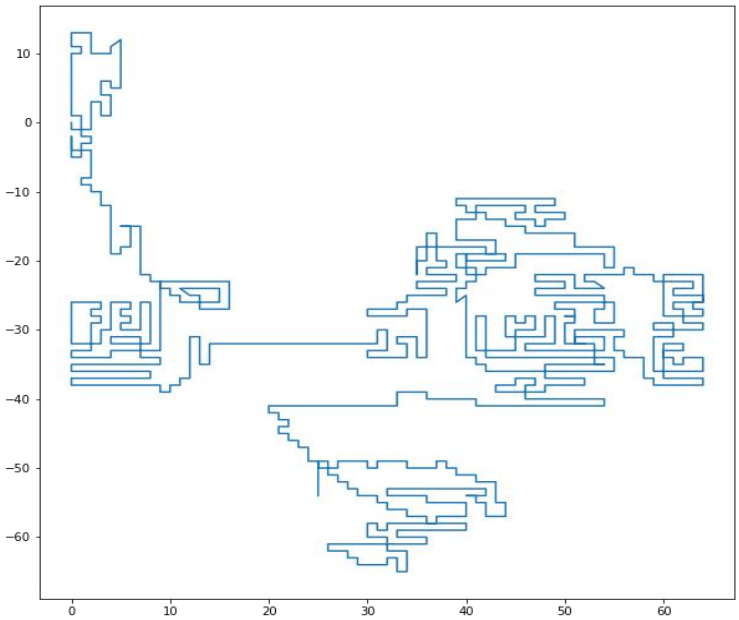
数据集

数据集包括机械臂和打印颜色等信息，旨在模拟圣诞卡制作的过程。数据集包含 N 个状态，其中每个状态表示一个特定的机械臂和打印颜色的配置。

做法

使用了 A*，其中 A* 算法使用启发式函数来估计从当前机械臂位置到目标机械臂位置的代价。我们在状态空间中进行搜索，并通过估计代价函数确定下一步的最优方向。

下图为结果（决策的轨迹）。



下一步计划

1、 平台能够完成一些离线任务
2、 将数据结果多样化展示
3、 可以通过深度学习算法挖掘相关数据的价值

4 第三次最终答辩 2023.6.13

4.1 搭建深度学习预测展示平台：

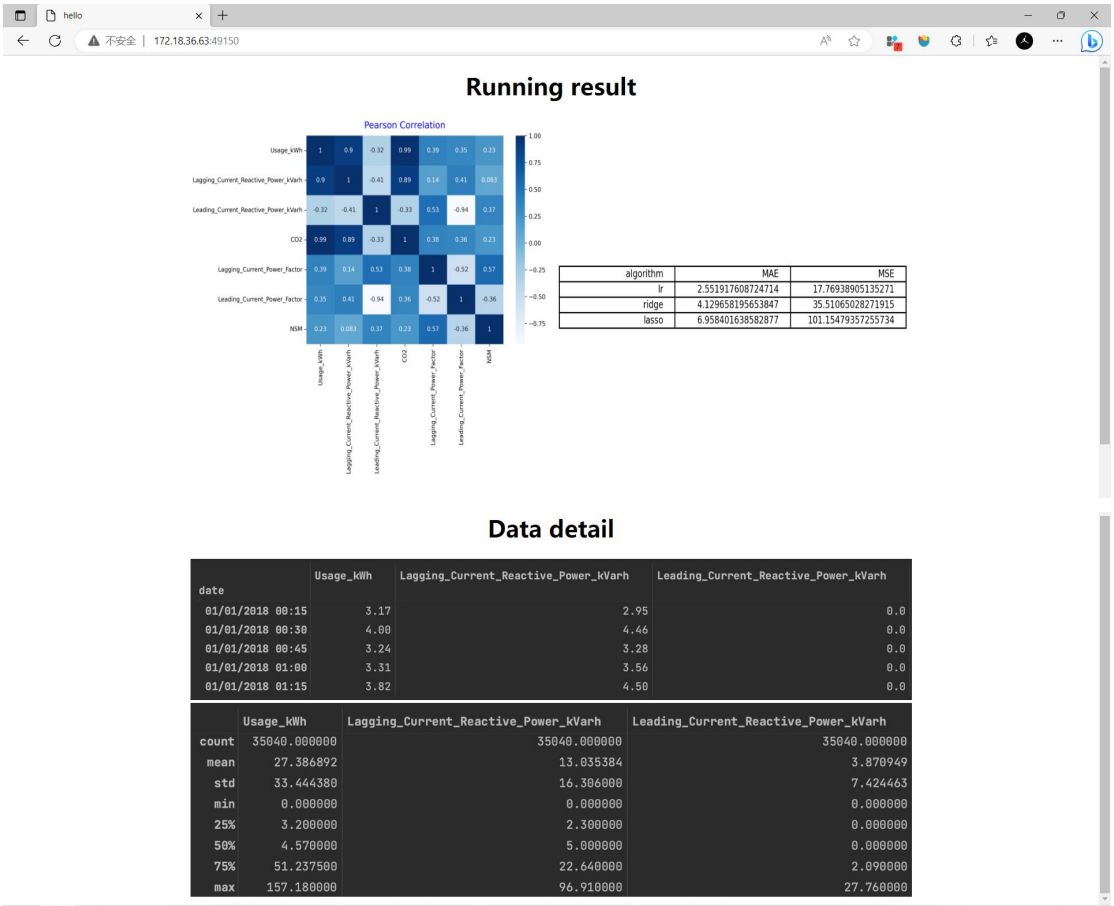
服务器端：

在成功通过 python 连接 Hive 数据库之后，着手将算法模型代码上传到服务器上，并将算法所需的数据写入到 Hive 数据库中，在运行算法模型时通过查询数据库获取数据。

同时服务器端通过编写自动脚本 run.sh 文件，可以实现通过运行脚本文件完成对算法模型的调用，在模型算法运行完成后会生成对应图像文件以及 csv 文件，通过 index.html 文件完成对运行结果的渲染，再通过 python 的 SimpleHTTPServer 库来将 index.html 文件所在目录变成一个建议服务器，并开放对应的 Docker 端口供外部访问。

以下是运行结果以及渲染效果：

```
root@55156866d345:~/algorithm# ./run.sh
===============================================
================Start==========================
===============================================
===========Server up at port 8080==============
root@55156866d345:~/algorithm# Serving HTTP on 0.0.0.0 port 8080 ...
10.26.225.40 - - [11/Jun/2023 09:30:37] "GET / HTTP/1.1" 200 -
10.26.225.40 - - [11/Jun/2023 09:30:37] "GET /model/Steel_industry_data_2/correlation.png HTTP/1.1" 200 -
10.26.225.40 - - [11/Jun/2023 09:30:37] "GET /model/Steel_industry_data_2/eps_table.png HTTP/1.1" 200 -
10.26.225.40 - - [11/Jun/2023 09:30:37] "GET /2.png HTTP/1.1" 200 -
10.26.225.40 - - [11/Jun/2023 09:30:37] "GET /1.png HTTP/1.1" 200 -
10.26.225.40 - - [11/Jun/2023 09:30:37] code 404, message File not found
10.26.225.40 - - [11/Jun/2023 09:30:37] "GET /favicon.ico HTTP/1.1" 404 -
```



date	Usage_kWh	Lagging_Current_Reactive_Power_kVarh	Leading_Current_Reactive_Power_kVarh
01/01/2018 00:15	3.17	2.95	0.0
01/01/2018 00:30	4.00	4.46	0.0
01/01/2018 00:45	3.24	3.28	0.0
01/01/2018 01:00	3.31	3.56	0.0
01/01/2018 01:15	3.82	4.50	0.0

	Usage_kWh	Lagging_Current_Reactive_Power_kVarh	Leading_Current_Reactive_Power_kVarh
count	35040.000000	35040.000000	35040.000000
mean	27.386892	13.035384	3.870949
std	33.444380	16.306000	7.424463
min	0.000000	0.000000	0.000000
25%	3.200000	2.300000	0.000000
50%	4.570000	5.000000	0.000000
75%	51.237500	22.640000	2.090000
max	157.180000	96.910000	27.760000

4.2 彻底完成 python 连接 hive 数据库：

在对 python 利用 pyhive 向 hive 数据库插入或改变数据前要对 hive 的权限进行预处理，否则只能读取不能改变数据：

修改 hive 配置文件 hive—site.xml：

```
<property>
  <name>hive.security.authorization.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hive.security.authorization.createtable.owner.grants</name>
  <value>ALL</value>
</property>
```

之后利用循环将钢铁数据插入数据库：

```
# 构造插入数据的SQL语句
sql = "INSERT INTO steel (date_column, usage_kwh, lagging_current_reactive_power_kvarh,
 leading_current_reactive_power_kvarh, co2_tco2, lagging_current_power_factor, leading_current_power_factor, nsm,
 weekstatus, day_of_week, load_type) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
values = (date_column, Usage_kWh, Lagging_Current_Reactive_Power_kVarh, Leading_Current_Reactive_Power_kVarh,
 CO2_tCO2, Lagging_Current_Power_Factor, Leading_Current_Power_Factor, NSM, WeekStatus, Day_of_week, Load_Type)

# 执行插入操作
cursor.execute(sql, values)
```

之后将数据读取并转化为 dataframe 类型：

```
from pyhive import hive
import pandas as pd

conn = hive.Connection(host='172.18.36.63', port=1145, username='root')

cursor = conn.cursor()

cursor.execute('SHOW DATABASES')
cursor.execute('USE mysqltest')
cursor.execute('SHOW TABLES')

cursor.execute('SELECT * FROM steel')
results = cursor.fetchall()
df = pd.DataFrame(results, columns=[desc[0] for desc in cursor.description])

print(df.shape)
# print(df)


cursor.close()
conn.close()
```

其中，用到了 pyhive 库来进行数据库的连接，由于 hive 数据库存于服务器上的 docker 容器中，所以需要提前做好端口映射。

客户端连接 Hive 需要使用 HiveServer2。HiveServer2 是 HiveServer 的重写版本，HiveServer 不支持多个客户端的并发请求。当前 HiveServer2 是基于 Thrift RPC 实现的。它被设计用于为像 JDBC、ODBC 这样的开发 API 客户端提供更好的支持。Hive 0.11 版本引入的 HiveServer2。另外用到的包有 pandas，用于将数据库查询到的数据转化为更加便于处理的 dataframe 类型。

之后便可以对数据进行各种算法的处理和分析。

4.3 钢铁数据分析：

对钢铁行业数据进行分析，并使用线性回归和正则化回归模型进行预测和评估。包括数据准备、相关性分析、数据预处理、模型训练和评估、以及在每个输出步骤导出相应的结果（例如表格、图片信息）。

1．数据准备
在开始分析之前，我们首先导入所需的库，并读取了一个名为"Steel_industry_data.csv"的 CSV 文件，将其存储在一个名为 df 的 DataFrame 中。然后，我们对一些列名进行了重命名，并保存了 DataFrame 的前几行数据(df.head())到"df_head.csv"文件中，以便查看数据的概览。

2．相关性分析
为了了解各个变量之间的相关性，我们绘制了一个热力图。通过使用 Seaborn 库的 heatmap 函数，我们可以直观地查看变量之间的相关性程度。相关性分析结果以热力图的形式保存在"correlation.png"文件中。

3．数据预处理
为了进行模型训练，我们进行了一些数据预处理操作。首先，我们将 DataFrame 中的'date'列设置为行索引，以便更方便地处理时间序列数据。然后，我们将分类变量进行了独热编码，以便在模型训练时使用。再次保存了处理后的 DataFrame 的前几行数据(df.head())到"df_head2.csv"文件中，以便查看预处理结果。

4．模型训练和评估
我们将数据集划分为训练集和测试集，并使用三种回归模型进行训练和评估：线性回归模型(Linear Regression)、岭回归模型(Ridge Regression)和 Lasso 回归模型(Lasso Regression)。对于每个模型，我们计算了模型在测试集上的决定系数(R2 score)。

5．评估指标
为了对模型进行更全面的评估，我们计算了两个常用的回归评估指标：平均绝对误差(MAE)、均方误差(MSE)。我们使用这些指标衡量了每个模型在测试集上的性能，并将结果以表格的形式展示出来。

6．结果展示
为了更直观地展示评估结果，我们使用 Matplotlib 库绘制了一张包含模型评估指标的表格，并将其保存为"eps_table.png"文件。

通过以上步骤，我们完成了对钢铁行业数据的分析和模型评估，得出了模型的性能指标以及导出了程序生成的相关数据信息，以供进一步研究、决策、以及可视化。

预期成果

搭建出工业大数据平台，并根据需求，构建芯片工业数据处理算法，并申请外观专利和软件著作权。

参考文献

[1]任磊,贾子翟,赖李媛君,周龙飞,张霖 & 李伯虎.(2022).数据驱动的工业智能：现状与展望. 计算机集成制造系统(07),1913—1939. doi:10.13196/j.cims.2022.07.001.

[2]许庆兵.(2022). 工业智能化背景下大数据的应用研究. 科技资讯(03),19—21.doi:10.16661/j.cnki.1672—3791.2111—5042—8751.

[3]朱佳乐.(2019).大数据背景下智能化工厂的建设. 科技创新与应用(23),80—81.

[4]周慧芝.(2022).浅析互联网大数据发展机遇及挑战. 智慧中国(08),71—72.

[5]罗亮.基于 Docker 的工业大数据平台持续服务关键技术研究.2019.浙江理工大学,MA thesis.

[6] CHANG H, HARI A, MUKHERJEE S, et al. Bringing the cloud to the edge; proceedings of the Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on, F, 2014 [C]. IEEE.

[7] JI C, LIU S, YANG C, et al. IBDP: An industrial big data ingestion and analysis platform and case studies; proceedings of the Identification, Information, and Knowledge in the Internet of Things (IIKI), 2015 International Conference on, F, 2015 [C]. IEEE.

[8] 宫夏屹, 李伯虎, 柴旭东, et al. 大数据平台技术综述 [J]. 系统仿真学报, 2014, 26(03): 489—96.

[9] 吴思炜, 周晓光, 曹光明, et al. 热轧 C—Mn 钢工业大数据预处理对模型的改进作用 [J]. 钢铁, 2016, 51(05):

[10] 刘学军, 李长云, 万烂军. 基于 Spark 的工业大数据处理可视化平台应用研究 [J]. 福建电脑, 2017, 33(12): 23—5

[11] Tom White, Hadoop: The Definitive Guide. O'Reilly Media, 2015.

[12] Edward Capriolo, Dean Wampler, and Jason Rutherglen, Programming Hive: Data Warehouse and Query Language for Hadoop. O'Reilly Media, 2012.

[13] 张文杰.基于工业物联网和大数据分析技术的设备故障智能预测系统研究.2021.郑州大学,MA thesis.