中信 EMR 智能数据平台

用户手册

日期：2017 年 7 月

目录

1 前言

概述

本手册为用户介绍了 EMR 产品的相关服务及使用方法。

使用对象

数据科学家、安装人员、数据开发工程师、运维工程师。

项目背景

为了构建高效可靠的云平台，同时提供可伸缩的弹性数据平台部署服务，从而支持中信集团高性能计算和大数据存储及应用，中信云网部署了 EMR 智能数据平台。

2 功能简介

新用户注册后，通过账户和密码登录 EMR 平台，进入平台主界面，系统默认为集群管理界面。

EMR 平台提供五项服务，包括：总览、数据平台、分析应用、模型示例、用户中心和管理中心。

功能介绍如下：

总览：为用户提供任务、模块及项目的数据总览。

数据平台：为用户提供集群创建和管理平台，可选择任意公有云服务类型部署 EDS 数据库服务。同时，为用户提供 EDS 数据库操作平台，支持完整的 ANSI SQL（SQL92/SQL99/SQL2003/OLAP Extension）查询语言，可进行轻量和高效的抽样数据分析功能，同时，适用于编写算法模块前期的探索工作。

分析应用：为用户提供全量数据的自助分析平台，用于导入或创建模块及项目工作流。

模型示例：为用户展示了机器学习相关算法模型和机器学习建模流程，同时支持用户在该平台根据需求新建机器学习模型。

用户中心：为用户提供基本设置及相关权限操作。

管理中心：对整个平台的日志显示。

3 数据平台使用指导

3.1 集群管理

3.1.1 新建集群

单击左侧导航栏，选择数据平台>集群管理，系统自动跳转到集群管理界面。单击“新建”用户可创建集群，用户根据需求填写相关信息，如所图 5-1 所示：

图 3-1 创建集群

📖说明：如果用户为私有部署请选择“类型”为“私有”，私有部署请联系我们工程师。

单击“下一步”，自动跳转到“添加”界面，用户根据需求选择区域、主节点类型、从节点类型、扩展从节点数量，单击“新建”如图 5-2 所示：

图 3-2 插入节点📖说明：在进行大量数据的批量导入导出时，如果源（目的）数据库和 EDS 处于不同网络环境，带宽上限和网络状况往往是制约导入速度的最主要因素，导入时间(S)>=数据总量(MB)/可用带宽(MB/s)。在网络质量较差时，也会出现导入(导出)超时或失败的现象。如果源（目的）数据库和 EDS 间网络环境稳定，甚或处于同一网络或者通过专线连接，导入（导出）速度也会受到两端服务器硬件配置（磁盘 I/O 性能等）影响。

3.1.2 集群管理

集群创建完成后，通过集群列表可查看集群序号、IP、名称、描述、生命周期类型、状态和创建时间等信息。单击操作下的编辑图标，可对集群名称和描述进行编辑。

图 3-3 集群管理界面

📖说明：集群状态包括：等待、创建中、启动中、运行中、重启中、停止中、已停止、删除中、已删除、未知、异常等 11 种状态。

单击某个集群序号，系统跳转到集群界面，可对该集群进行运维、停止、启动和权限设置等操作，可弹性释放资源，优化资源利用率，如图 5-4 所示。

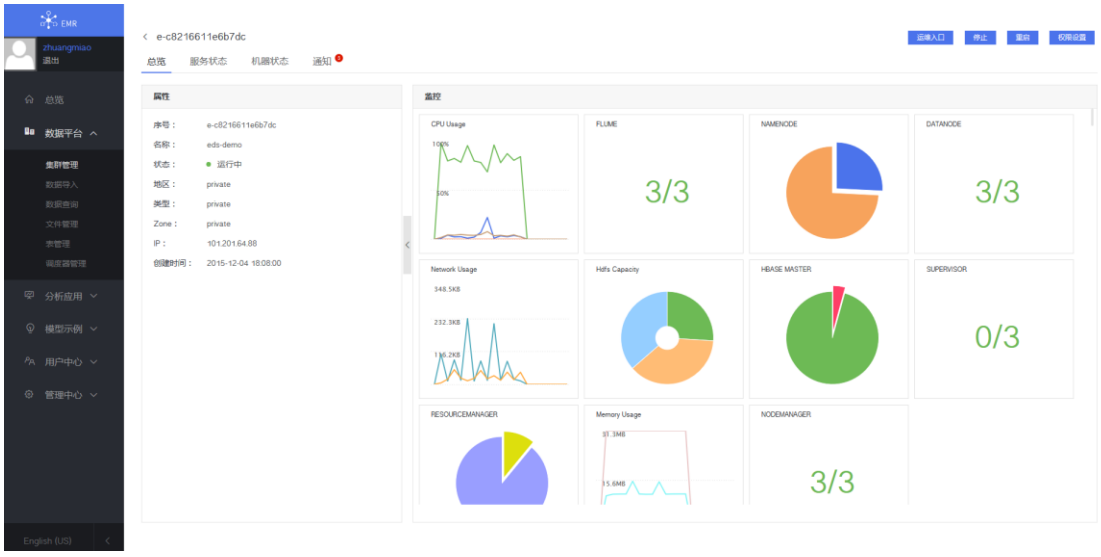


图 3-4 查看集群

3.2 数据导入

数据导入支持多种类型的数据源导入。比如：关系型数据库（MySQL/Oracle 等）、NoSQL 数据库（Mongodb/Cassandra 等）、文件（excel/csv/ftp 等）。如下图，

用户还可以查看已导入的数据信息。单击名称，可查看导入详情。单击目标数据，可查看导入的数据库的表字段及表数据。封装则为用户提供了进一步管理数据资产的功能，可将导入的数据封装成模块直接在以后的分析中应用。

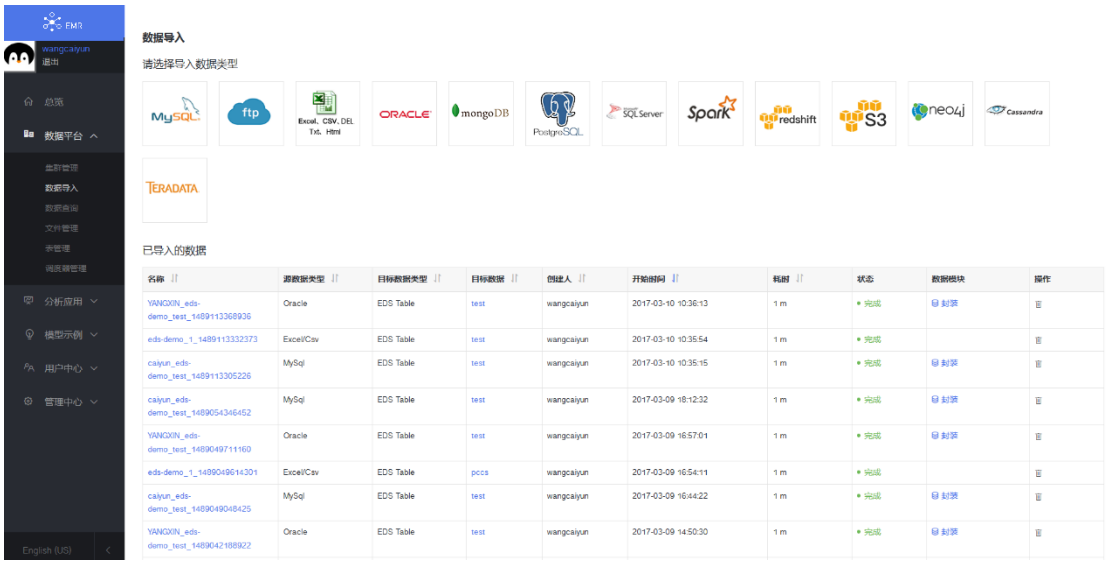


图 3-5 数据导入

导入 MySQL 数据库示例：

单击 MySQL 图标，可导入 MySQL 类型数据库。

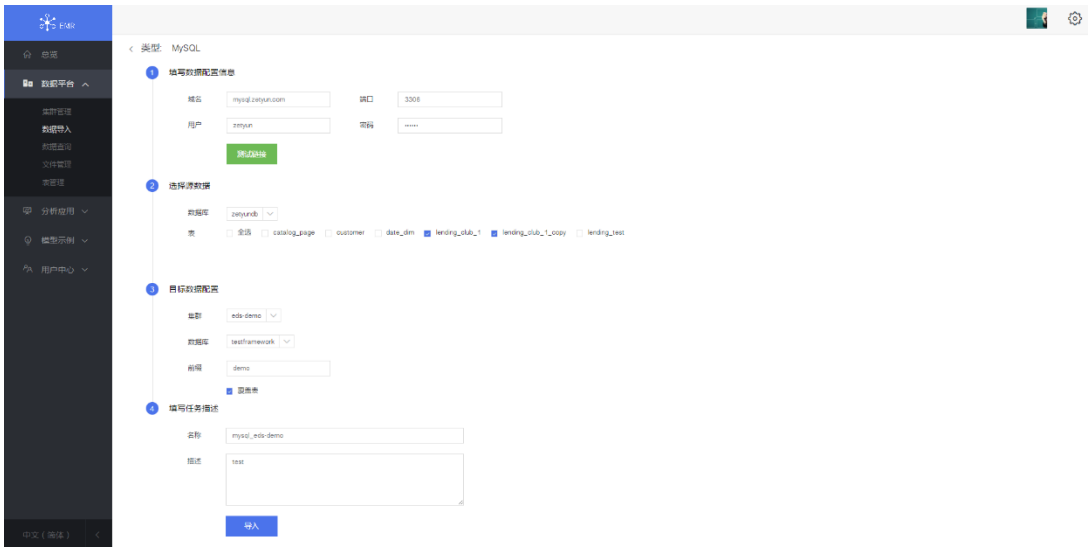


图 3-6 导入 MySQL 数据库

数据配置信息、源数据、目标数据、任务描述填写完成后，单击导入，系统会自动跳转到表管理界面，可查看刚刚导入的数据。

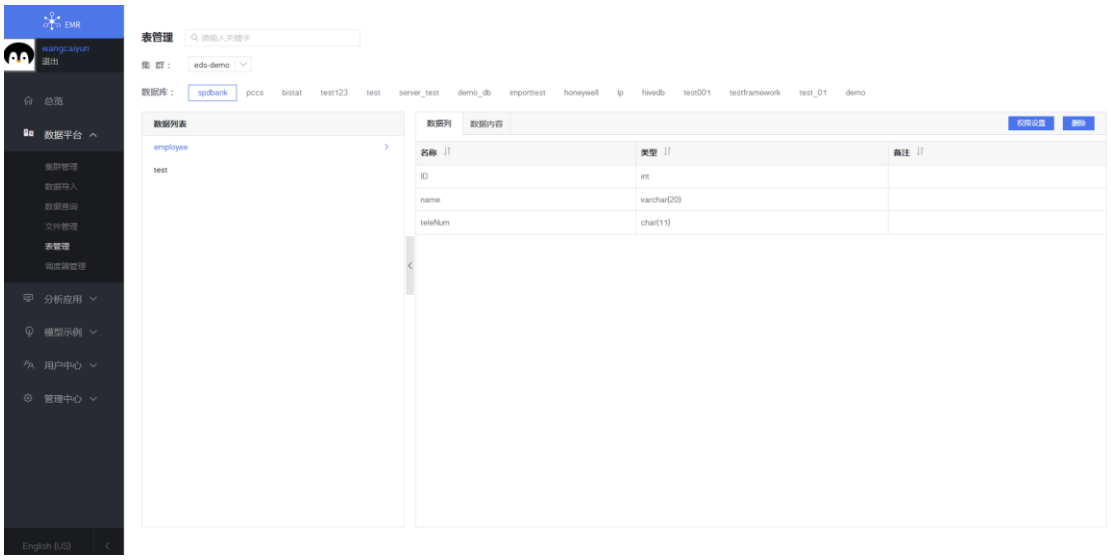


图 3-7 查看导入的数据

3.3 数据查询

“数据查询”为用户提供了数据库可视化操作的平台界面，同时，提供可视化操作 EDS 数据库和轻量、高效的抽样数据分析功能，支持 SQL 语句操作数据库，支持数据的导入、导出和分析等操作，适用于编写算法模块前期的探索工作。用户可以在代码区中通过 SQL 查询、shell、Distcp 等多种形式在不同 EDS 集群上的远程执行和实时交互，同时，数据查询内置数据可视化，支持自定义可视化类型。

查询命令示例：

查询集群 eds-demo 中 spdbank 中的数据。系统快速执行命令并展示结果，同时，用户可通过添加新视图自定义结果的展示形式。

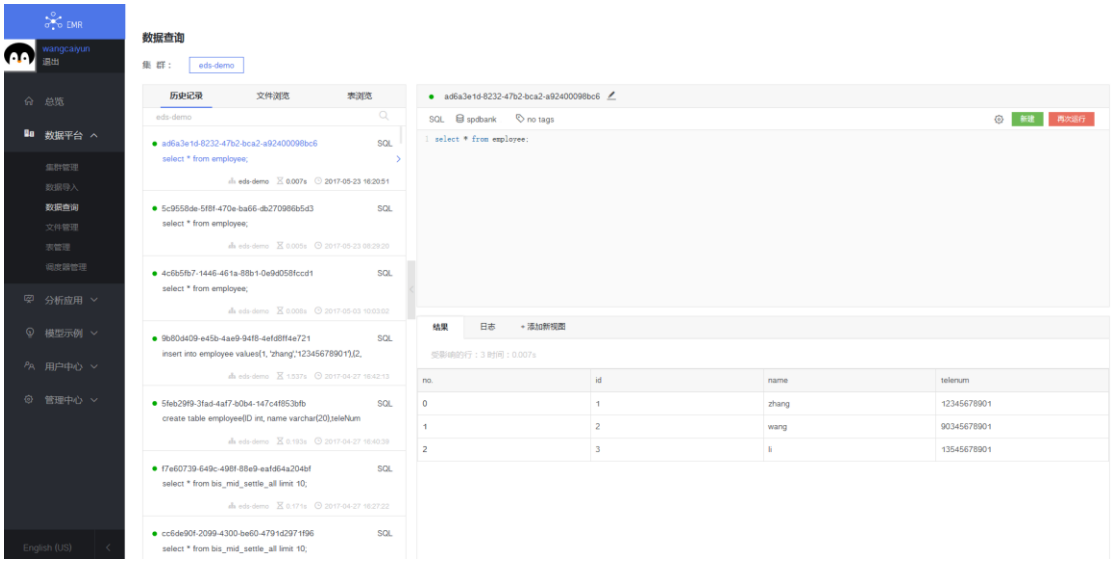


图 3-8 数据查询

3.4 文件管理

文件管理页面可以查看和管理用户所创建集群的 HDFS 文件。

页面支持从本地上传文件，新建文件夹。

🕮说明：上传文件和新建文件夹时需要进到 home 目录或者在 share 文件夹下进行。

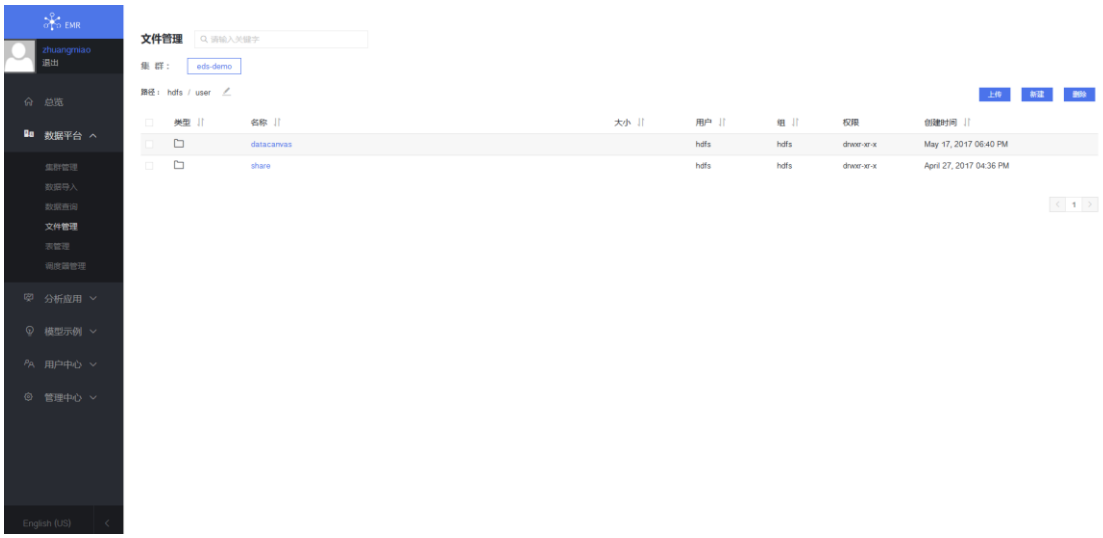


图 3-9 文件管理

3.5 表管理

表管理支持对集群内的数据库进行查看和管理。同时支持对数据库内的表格

进行权限设置，包括对某个表进行用户、用户分组以及读取、创建、编辑、删除等权限设置。

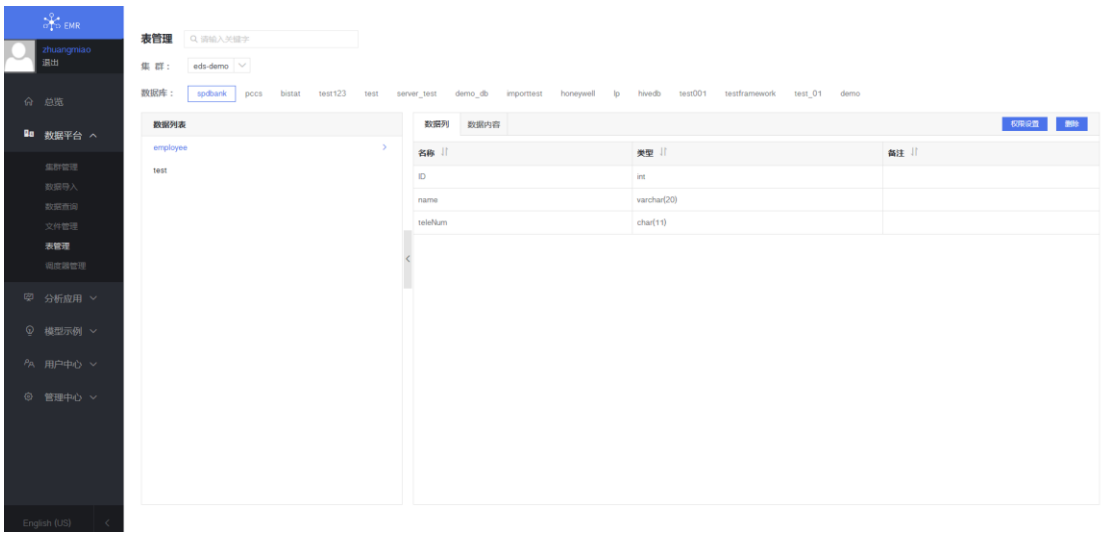


图 3-10 表管理

3.6 调度器管理

用于管理数据平台的集群资源，可创建不同的队列并分配资源大小和最大任务数等参数，用户在提交任务时指定某个队列就会使用这个队列的资源运行任务，非常直观的观察集群资源使用情况以及正在运行的任务。

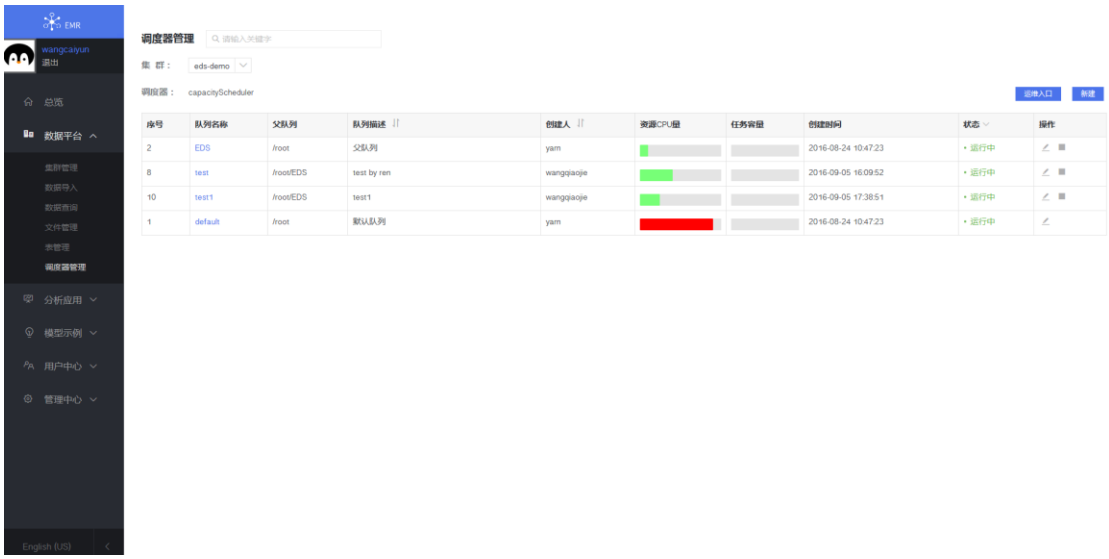


图 3-11 调度器管理

4 分析平台使用指导

4.1 数据平台功能简介

- 项目：实现某种具体商业需求的数据分析流。
- 模块：实现数据操作的最小单位，包括数据模块和分析模块，有可能是数据的导入导出；可能是针对数据库的某种操作，或者是使用某种机器学习算法对数据的处理和运算。
- 交互式探索：使用户能够方便直观的进行数据探索，高效实用机器学习算法进行分析，实时呈现分析结果，从而帮助用户解决实际业务中的问题。
- 算法定制：支持通过 R、Python、Hive、Pig 等进行轻量级的模块开发和编辑。
- 任务列表是项目运行后的结果列表。
- 可视化：系统可以提供饼图、柱状图、折线图等常见图形来满足不同的展示和分析需求。
- API 发布：对数据或算法模型进行封装，以 RESTAPI 的方式提供外部调用接口。

4.2 数据模块管理

EMR 内置部分开源模块库，用户可直接使用，模块功能可参见模块下方描述内容。

单击左侧导航栏分析应用>数据模块，系统跳转到数据模块界面，可对数据源模块进行查看和新建，通过点击上方分类标签进行模块筛选。

单击模块名称，可查看数据模块详情、历史记录、引用记录等，并支持对该模块进行评论。

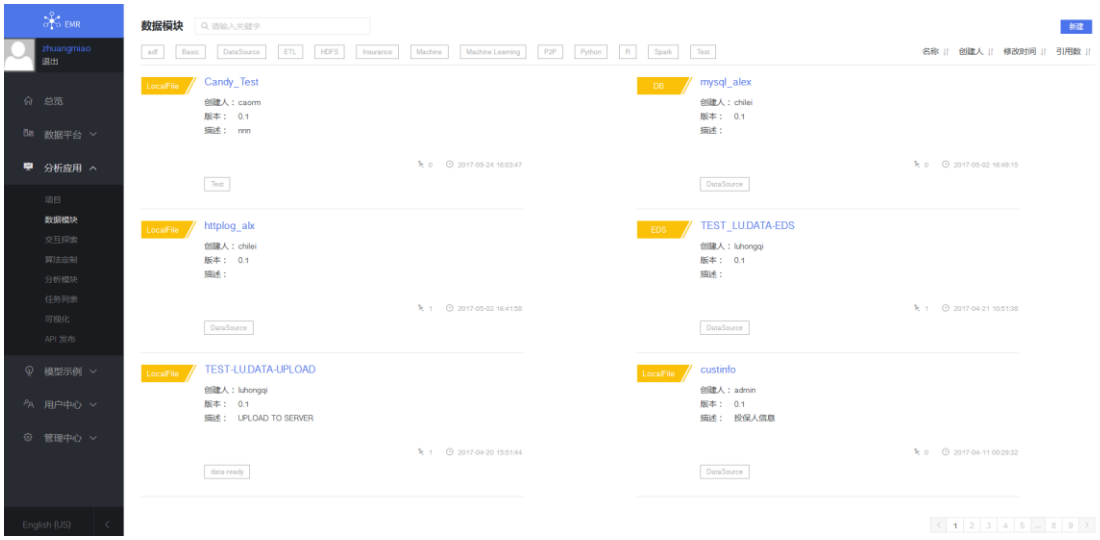


图 4-1 查看数据源模块

单击“新建”，根据自身需求创建数据源模块。

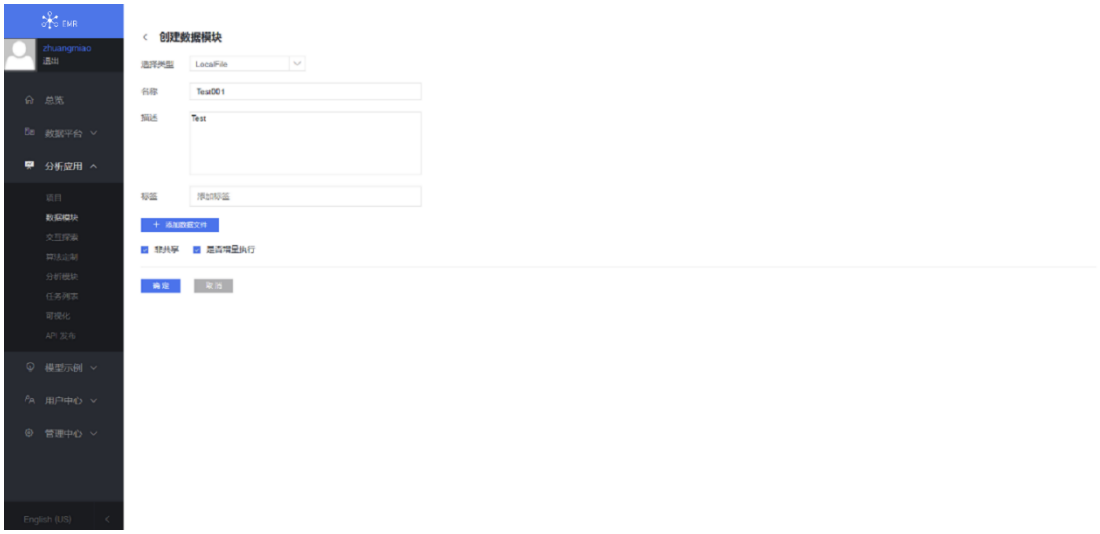


图 4-2 新建数据源模块

📖说明：数据源类型支持 LocalFile/Http/Ftp/AWS S3/HDFS/HIVE/DB/EDS。

4.3 分析模块管理

单击左侧导航栏分析应用>分析模块，系统跳转到分析模块界面，可对数据源模块进行查看和新建，通过点击上方分类标签进行模块筛选。

单击模块名称，可查看数据模块详情、历史记录、引用记录等，并支持对该

模块进行评论与下载。

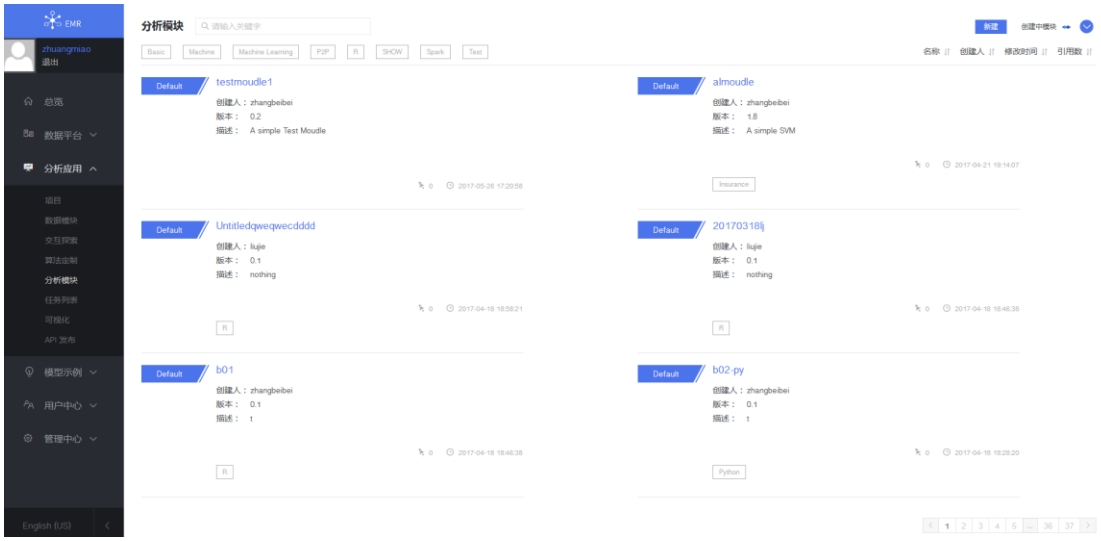


图 4-3 查看分析模块

4.3.1 编辑模块

单击模块可编辑模块并查看模块详情、历史记录和引用记录等信息，如图 6-4 所示。单击标签编辑图标，可以修改、新增标签信息。

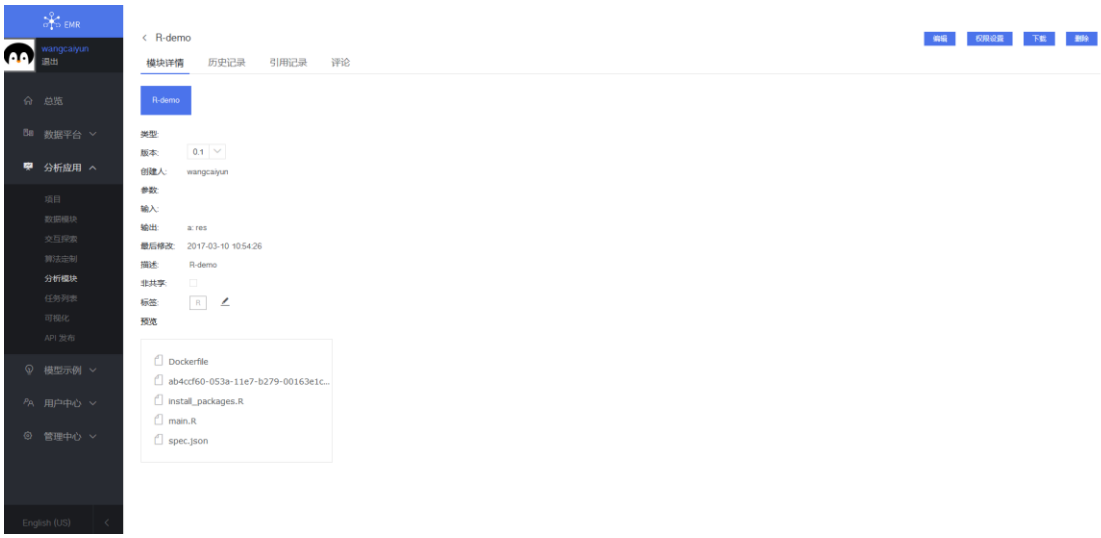


图 4-4 编辑模块

📖说明：

- 通过页面新建的模块支持编辑；通过 Screwjack 工具编写的模块暂不支持在界面进行编辑。

- 用户只能对自己创建的模块进行编辑、权限设置和删除等操作，其他用户创建的模块仅拥有查看权限。

4.3.2 新建模块

EMR 支持用户自定义模块，包括以下两种方式：

- EMR 支持在页面上简单快速的进行轻量级模块的开发和编辑。
- EMR 提供了专门的模块开发工具 Screwjack，用于复杂脚本的编写、测试及导入，单击界面右上方帮助问号可获取 Screwjack 工具包及使用方法。

(备注：Screwjack 已在 Github 中开源，地址：https://github.com/DataCanvasIO/screwjack)

下面以创建分析模块为例：

进入分析模块界面，单击页面右上方“新建”按钮，开启创建模块引导，帮助用户快速定义模块的输入、输出和参数配置。模块生成后将自动上传至 EMR 模块库，供工程调用。

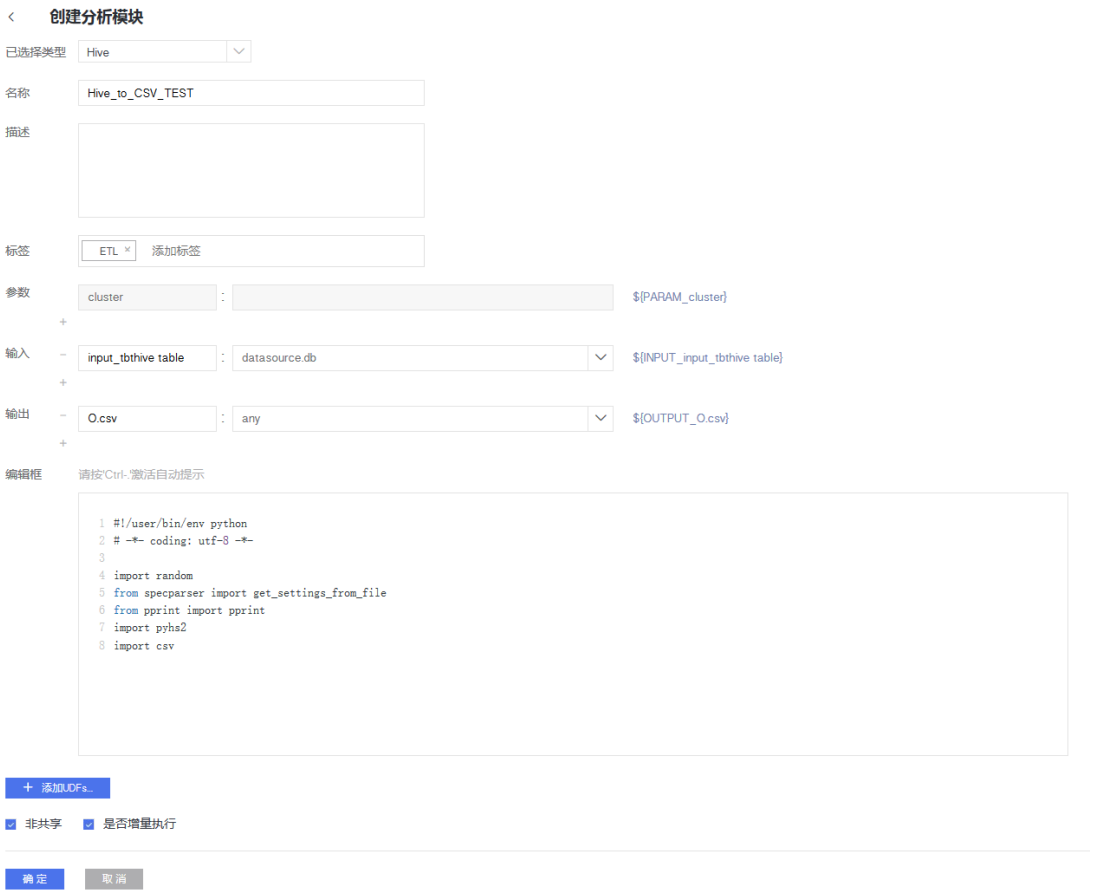


图 4-5 创建模块

📖说明：页面仅支持 Hive/Pig/Hadoop_Jar/Python/SCI_Python/R 语言编写轻量级模块，在“选择类型”下拉框中可见。如需采用其他语言编程请使用 Screwjack 工具，工具包获取及使用方法参加界面右上方帮助问号。

4.4 项目管理

单击左侧导航栏“项目”，页面跳转到“项目”页签，可查看、导入和创建项目信息。EMR 内置部分项目工程，用户可直接使用。

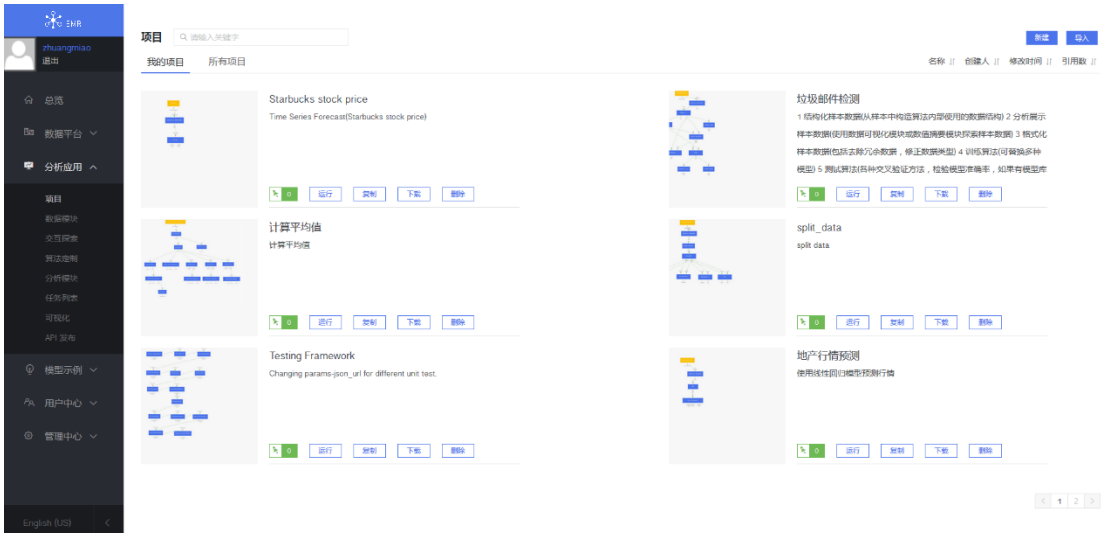


图 4-6 项目列表

单击项目画布，可查看当前项目版本号及模块信息，也可以根据需求编辑项目。单击页面右上方更多按钮，弹出下拉框，可进行如下操作：

- 创建项目：创建新的项目工作流。
- 打开项目：用户可打开其他项目流程，可选择是否保存当前项目。
- 变量设置：用户可设置全局变量。
- 数据环境：用户可选择数据环境，数据环境为全局变量的集合。
- 编辑项目：可对项目名称、描述进行编辑，支持取消共享。
- 复制项目：可对本项目进行复制。
- 权限设置：支持对本项目的用户、用户分组及读取、创建、编辑、删除等权限进行设置。
- 引用记录：用户可查看项目运行记录。

4.4.1 编辑已有项目

用户可根据自身需求编辑项目流程，编辑完成保存后，系统会自动更新版本号。

📖说明：用户需提前获取该项目的编辑权限。

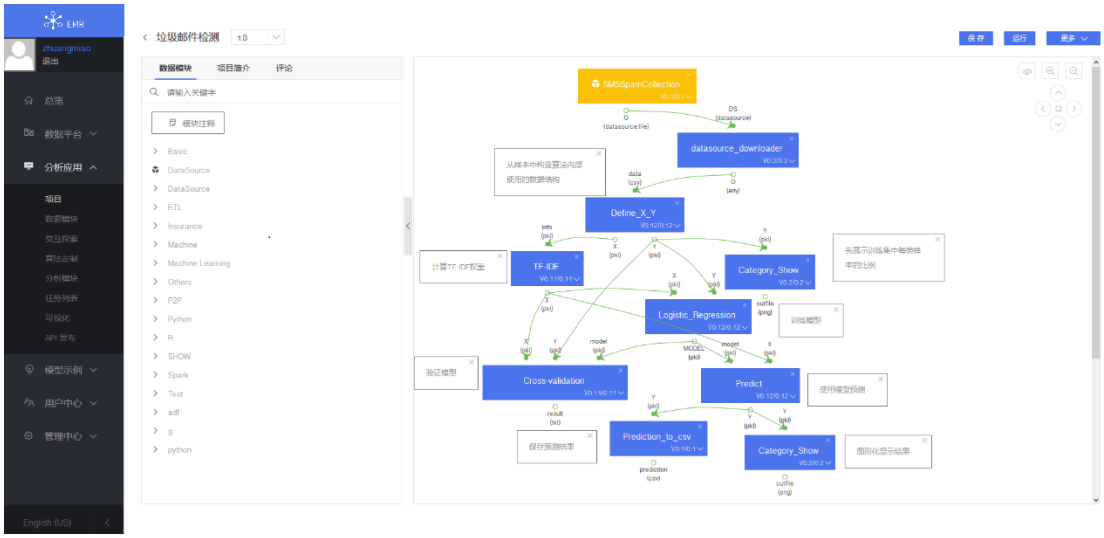


图 4-7 编辑项目

双击项目中的模块，用户可对模块配置进行编辑，保存后系统自动更新版本号。

图 4-8 编辑模块配置

4.4.2 新建项目

单击“新建”，用户可根据需求新建项目。名称和描述填写完成后，单击“新建”，页面跳转到创建项目页面，如图 4-9 所示：

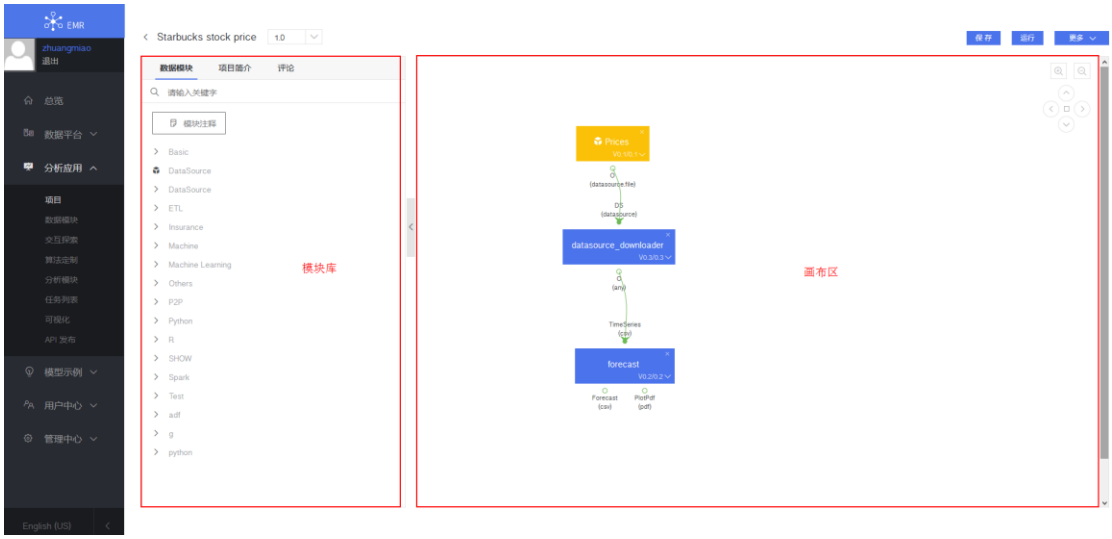


图 4-9 创建项目

用户将鼠标置于“第一个模块”处可弹出加号，通过单击加号，系统自动弹出模块下拉框，下拉框中模块已自动匹配上级模块的输出类型，便于用户选取。

同时，平台支持直接从左侧模块库中拖拽模块至工作区域，在确保模块输入类型与上级模块的输出类型相匹配的情况下将功能模块通过数据耦合的方式连接在一起，形成分析工作流。

4.4.3 创建任务

流程创建完成后，单击保存，系统自动生成版本号，单击“运行”弹出如下对话框，用户可自定义任务运行模式。

图 4-10 创建任务

单击运行后，自动跳转到任务界面，查看当前项目运行状态。

4.5 交互式探索

交互式探索可作为用户创建分析模块的前期探索工作；对于在“交互式探索”中有价值的查询或算法，通过控制台的算法定制，用户均可以将其固化为模块，经过简单的页面拖拽即可实现数据的清洗、挖掘、分析和可视化，并托管在分析平台之上进行工作流的控制以及定期的调度。

图 4-11 交互式探索呈现

交互式探索具备以下优势：

- 交互式数据探索，即刻获取分析结果，可多任务同时运行；
- 内置常用的数据分析和机器学习算法，可直接调用运行，方便数据团队协同与共享；
- 支持多种分析语言，Scala(使用 Apache Spark)、Python(Apache Spark)、Spark SQL、Hive、Markdown、Shell 等等；使用者可选择自己擅长的任意一种语言进行分析探索和挖掘，提高了数据分析的舒适度，降低数据分析的语言门槛。
- 内置多种可视化效果，使用者可选择最优展现形式。

4.6 算法定制

算法定制是指用户可以选择 Hive、Pig、Hadoop_Jar、Python、SCI_Python 和 R 语言来编写轻量级应用分析模块。

在左侧导航栏中选择“应用分析->算法定制”，点击您想使用的开发语言的图标，然后定义模块的输入、输出等参数配置，点击“创建模块”按钮创建新的模块。模块生成后将自动上传至 EMR 模块库，供工程调用，且会显示在<算法定制>页面的下方列表中。以 R 语言为例：

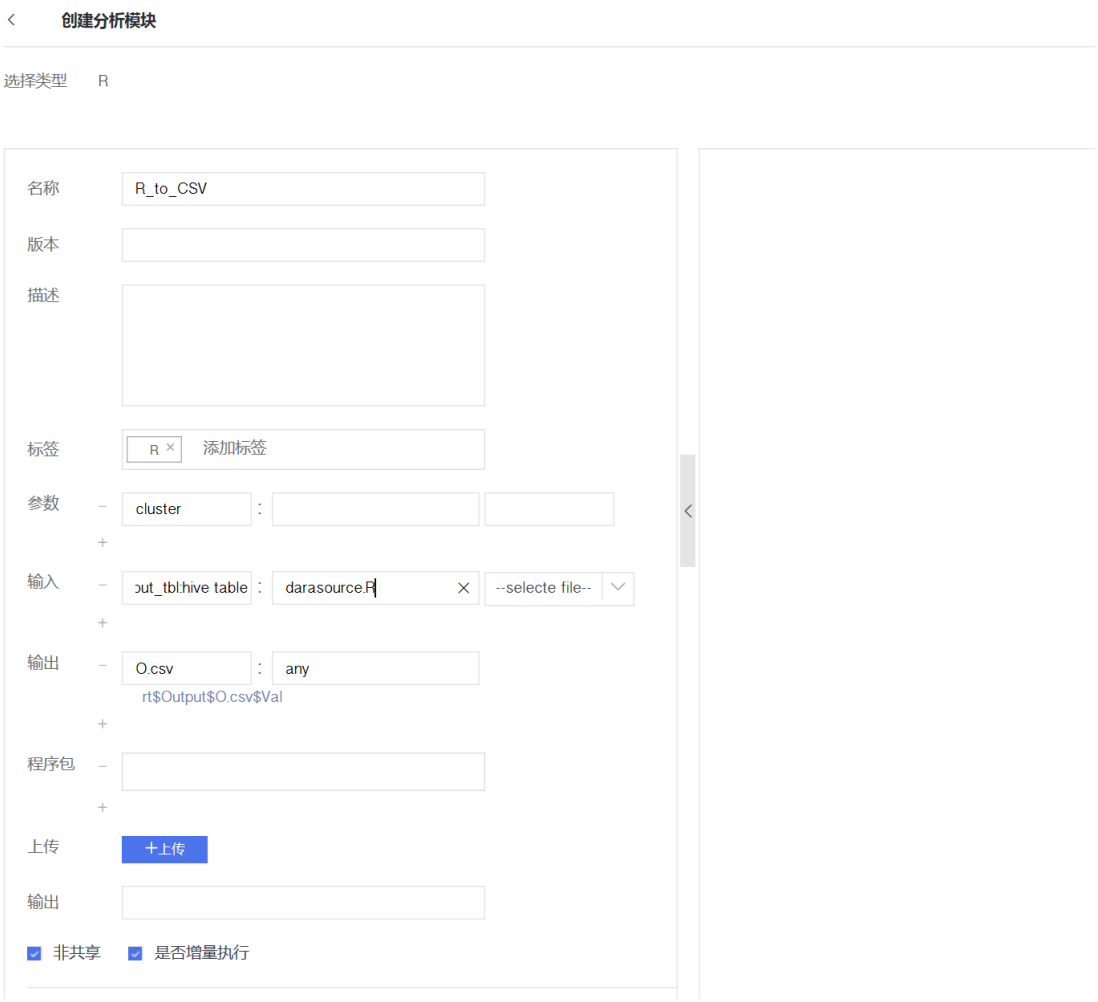


图 4-12 创建分析模块

4.7 查看任务

单击“任务列表”页签可查看运行后的全部任务，任务状态分为：等待、运行、异常、已暂停、已终止、完成。

📖说明：任务异常可能是链接超时网络中断或者模块逻辑自身错误，请根据以上原因进行排查。

单击任务名，可查看任务的流程图、模块列表、变量配置和运行日志。

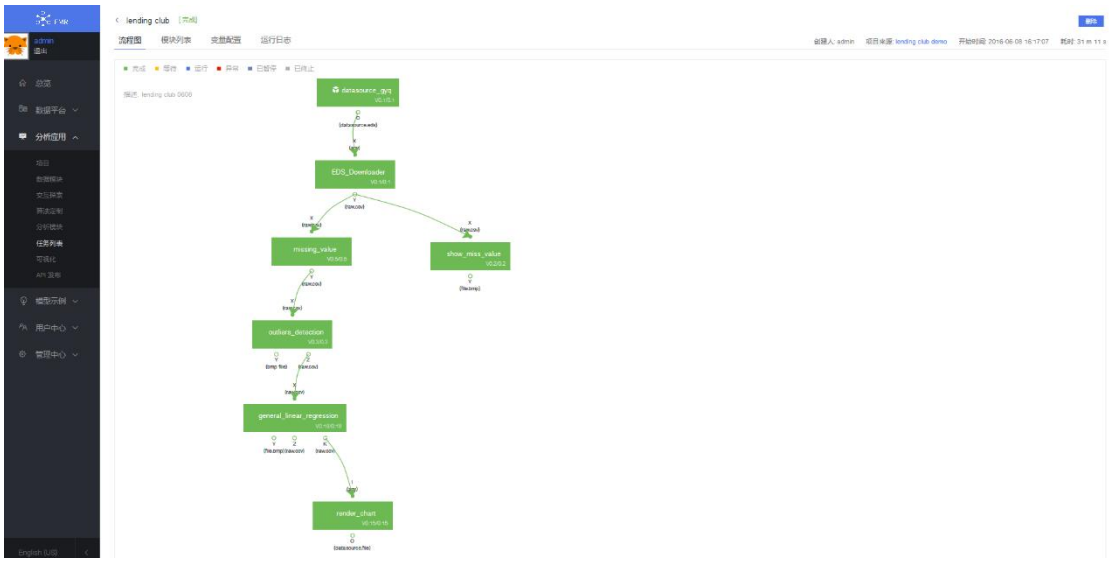


图 4-13 查看任务

任务流程图中，双击模块可查看当前模块的运行结果和日志，便于定位问题。双击结果输出模块，查看项目运行的可视化结果。

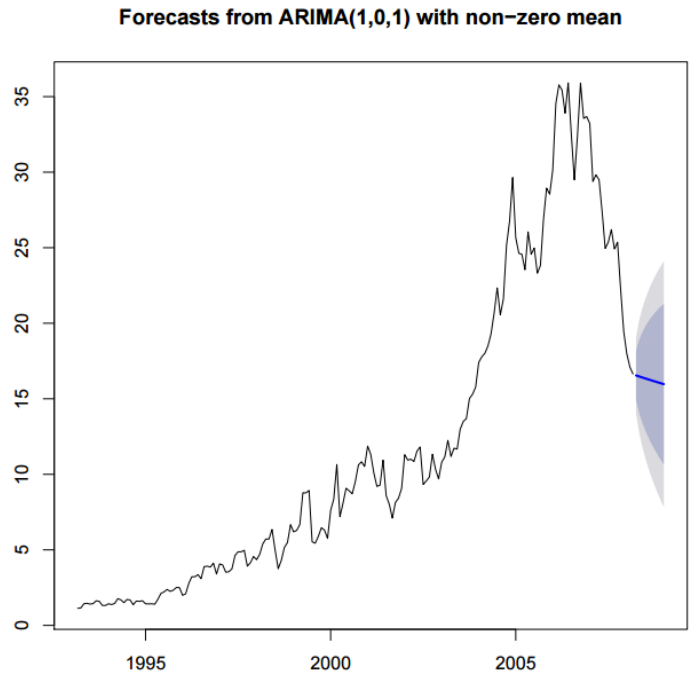


图 4-14 任务的可视化结果

4.8 可视化

系统可以提供饼图、柱状图、折线图等常见图形来满足不同的展示和分析需

求。

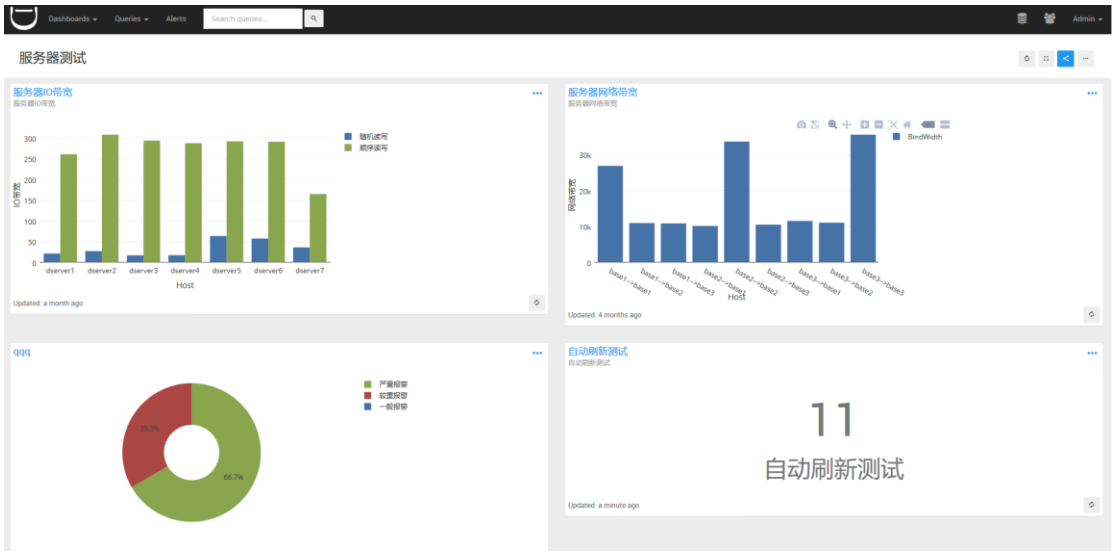


图 4-15 可视化

4.9 API 发布

对数据或算法模型进行封装，以 RESTAPI 的方式提供外部调用接口。

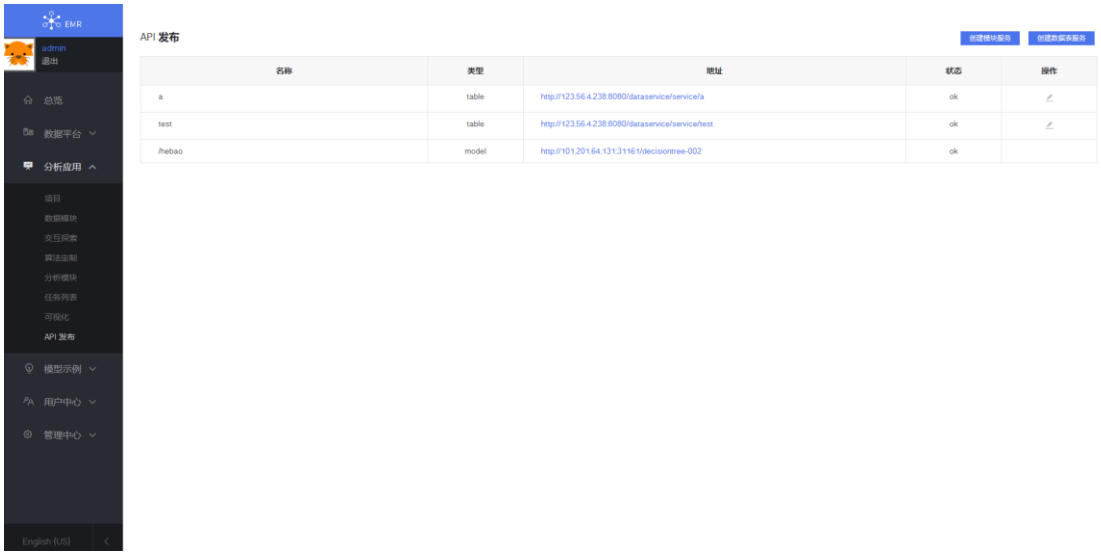


图 4-16 调用 API

5 模型示例

5.1 深度机器学习

可进行深度机器学习的探索工作。

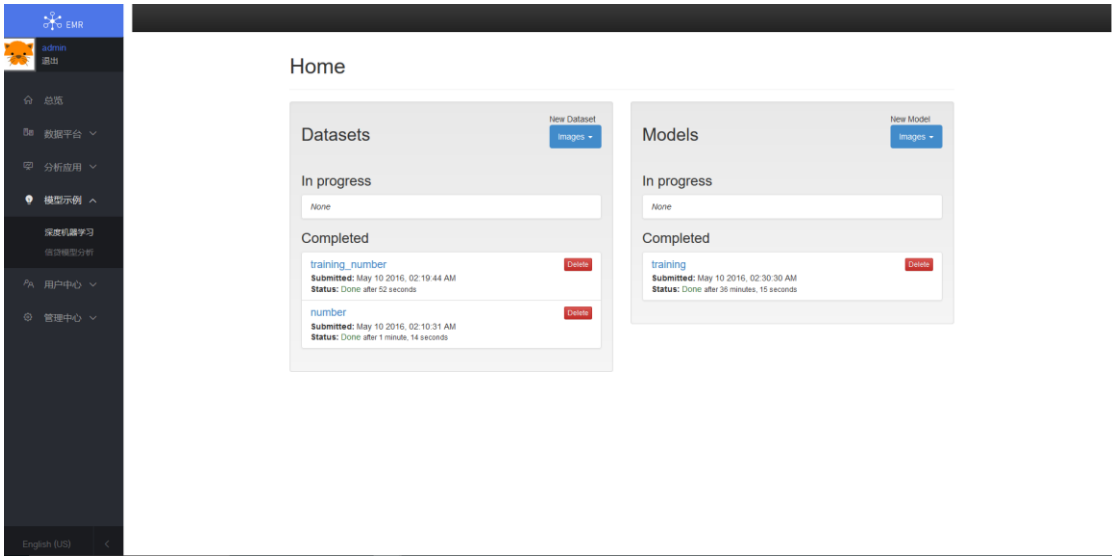


图 5-1 深度机器学习

5.2 信贷模型分析

信贷模型分析

界面展示了机器学习建模流程，列举了机器学习建模分析结果的可视化展现形式，并对机器学习典型模型做了清晰的图文介绍和模拟应用，旨在帮助用户快速了解机器学习算法，并通过典型模型的模拟应用使用户感知机器学习建模分析中的参数选择和逻辑思路。

EMR 以信贷风险评估模型为例，通过“用户输入申请者特征信息”填写参数；“数据结构总览”可以查看数据源并选择模型参数；“数据预处理和素描图”“分类百分比和热能图”“3D 关系图”为数据分析图形化界面，用户可通过更换变量、增减参数感知图形变化，了解数据趋势；“逻辑回归分析和模型汇总”“决策树模型分析”“随机森林模型”“神经网络模型”“支持向量机模型”分别介绍了各类模型的分析算法及公式，并对分析结果进行对比，帮助用户深入了解机器学习算法，并通过对比做出最佳判断。

图 5-2 建模页面

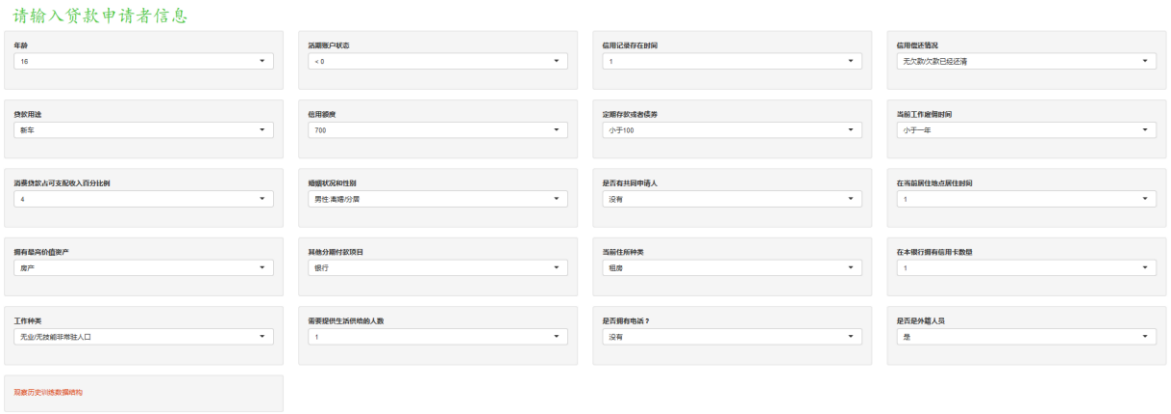


图 5-3 输入模型信息

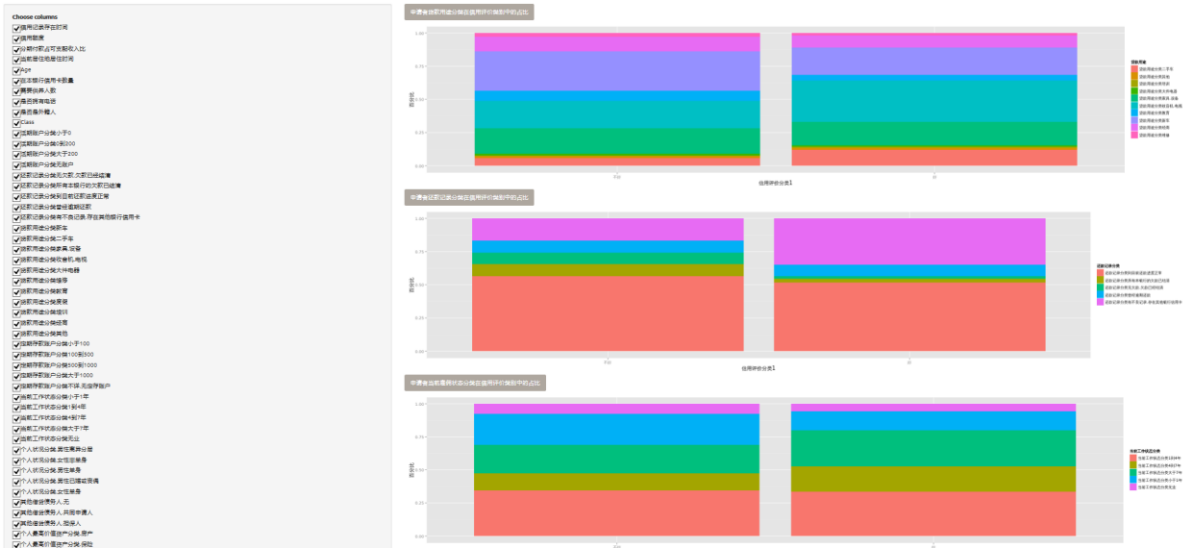


图 5-4 分类百分比和热能图

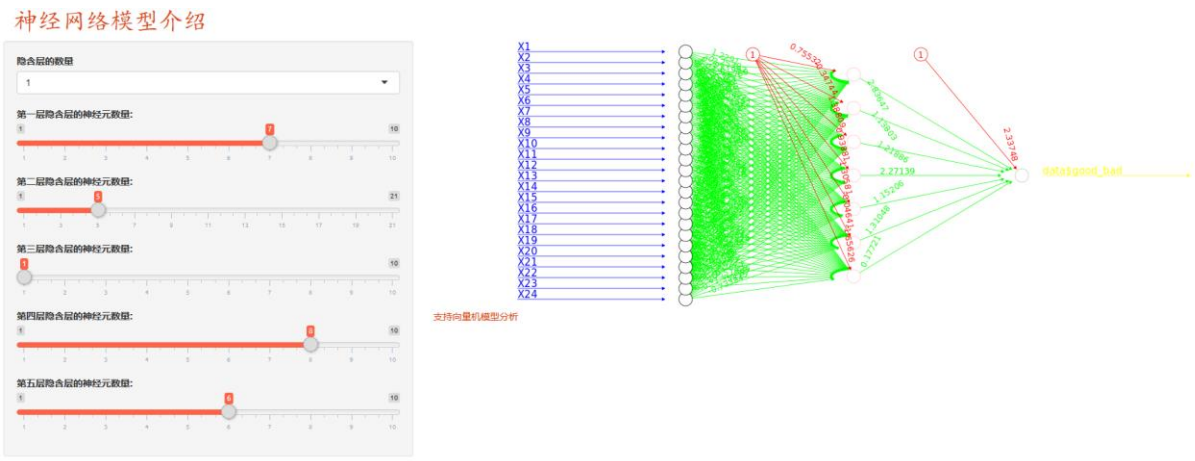

神经网络模型介绍
隐含层的数量
1
第一层隐含层的神经元数量:
第二层隐含层的神经元数量:
第三层隐含层的神经元数量:
第四层隐含层的神经元数量:
第五层隐含层的神经元数量:
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
X13
X14
X15
X16
X17
X18
X19
X20
X21
X22
X23
X24
2.33738
2.27139
dataset_ball
支持向量机模型分析

图 5-5 神经网络模型

6 用户管理

单击页面左上方功能按钮，选择“管理”，跳转到“管理”页面，用户可对个人资料、头像和密码进行修改、查看 Token，创建二级用户，并支持用户权限管理。

6.1 权限管理

EMR 支持多种不同的权限，用户根据需求加入不同的管理组同时被赋予相应的权限。

二级用户

管理员用户拥有创建二级用户权限，二级用户仅能对平台进行权限范围内的操作，如需更多操作需要向管理员用户申请增加权限。

用户分组

在某一分组内的用户仅能进行该分组权限分为内的操作。平台预置部分用户分组，同时支持新建。

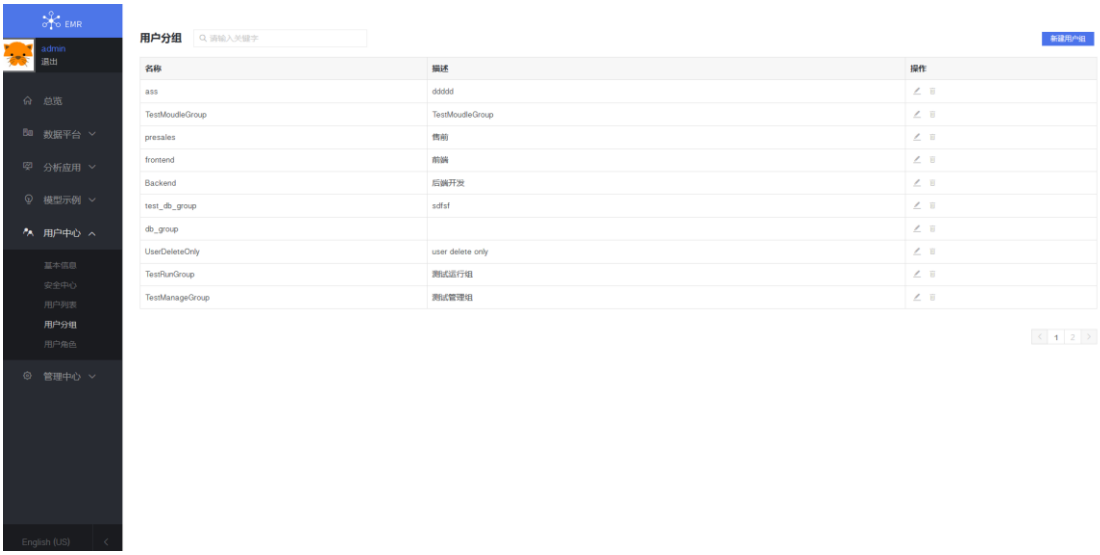


图 6-1 用户分组列表

单击“操作”下的“编辑”按钮，可以对用户角色进行编辑。

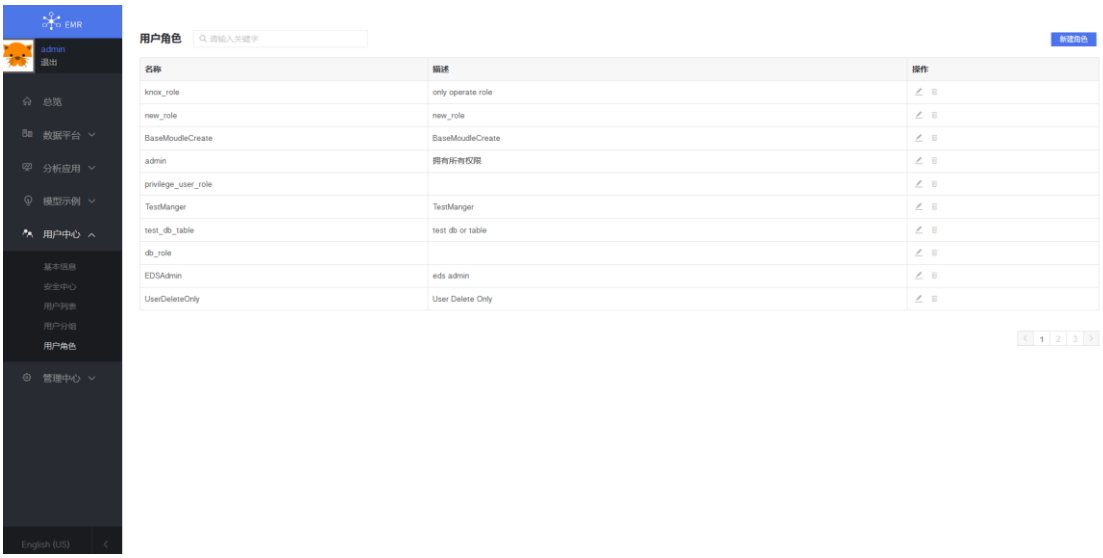


图 6-2 用户角色

用户角色

在某种用户角色内，被赋予了相关权限；平台预置部分用户角色，同时支持编辑和新建。

单击“操作”下的“设置”按钮，可对应用权限和集群权限进行编辑。

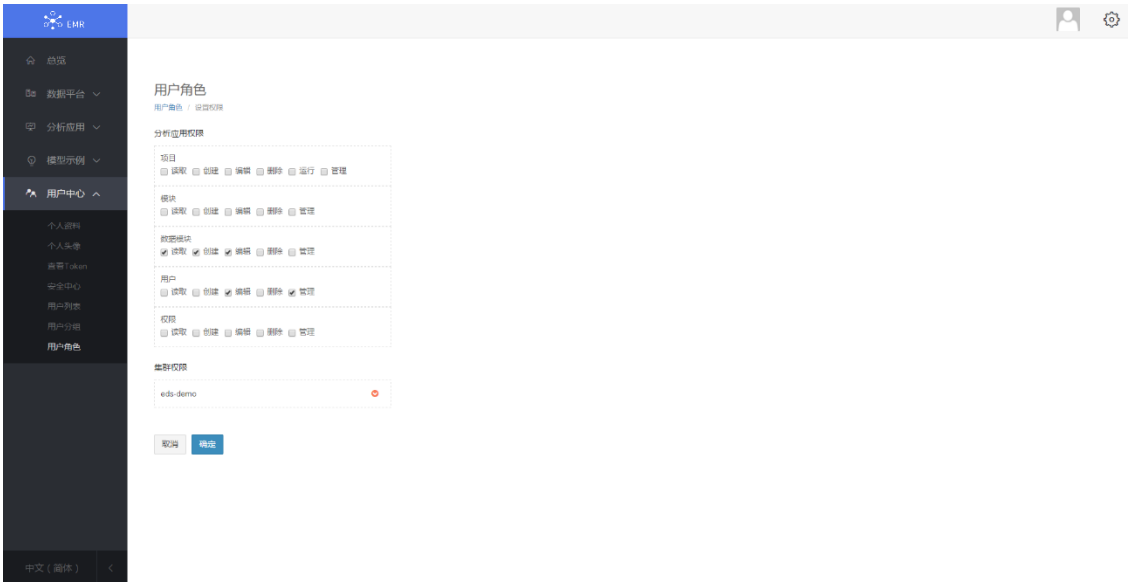


图 6-3 设置用户权限

7 管理中心

7.1 日志审计

Admin 用户拥有的权限，用来查看和记录子用户对平台进行的所有操作，发生问题便于追溯。

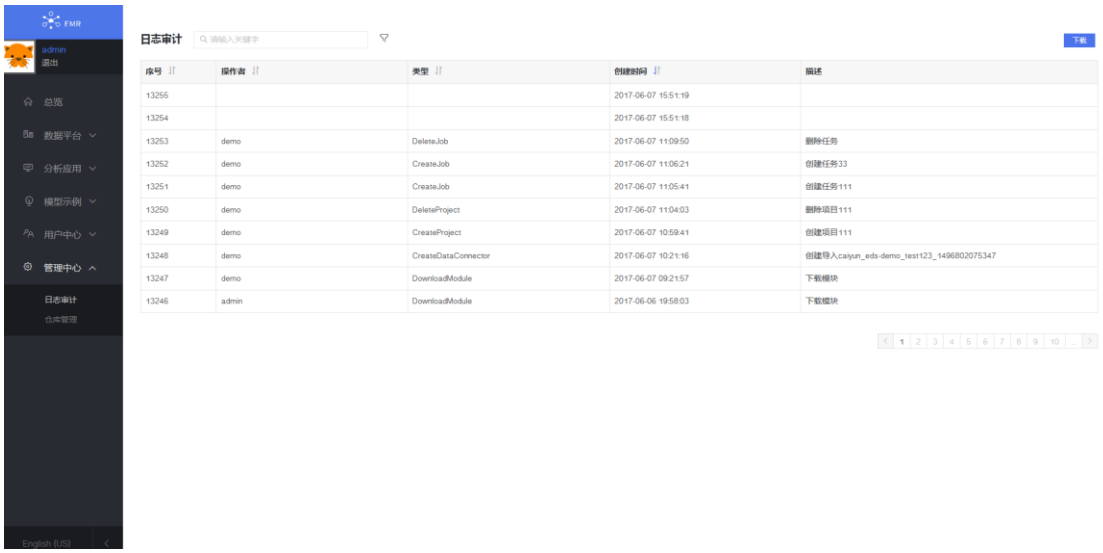


图 7-1 日志列表

7.2 仓库管理

R、Python 等主流的大数据分析语言提供了很多包做数据分析。除了给我们提供一个非常好的界面以便于我们进行统计分析以外，它们最大的优点尤其是 R 语言得到了全球开发者和许多数据科学大师们的鼎力支持。现在，可供世界各地的使用者下载的 R 包多达 7000 个。

除了一些大家熟悉的 R 包，比如 caret、ggplot、dplyr、lattice，还有很多被证实做数据分析很有用但是不易被我们察觉的包。鉴于此，我们创立了一个与数据分析相关且易于理解的语言包清单。

为了使这份向导更有参考价值，我们将这些算法包映射到我们的平台进行预建模、建模以及再建模的操作，便于内网的调用。


R
Python
SCI_Python
Hive
Pig
Hadoop_Jar
请输入名称
新增程序包
名称
描述
日期时间
暂无数据

图 7-2 语言包列表

8 附录：SCREWJACK

ScrewJack 是一个命令行式模块处理工具，用于复杂脚本的编写、测试及导入，旨在帮助研发人员快速完成模块的开发。

如果之前从未使用过 screwjack ，可以阅读相关章节快速入门。

Hierarchy of base images

输入/输出类型

为什么需要类型？

8.1 简介

8.1.1 基本概念

Screwjack 能够帮助模块设计者快速建立模块。模块是由一个 spec.json 的文件来定义的， 下面是一个 spec.json 的例子：

```
{
    "Name": "SVM",
    "Description": "A simple SVM",
    "Version": "0.1",
    "Cmd": "/usr/bin/python main.py",
    "Param": {
        "C": {
            "Default": "",
            "Type": "string"
        }
},
    "Input": {
        "X": ["csv"],
        "Y": ["csv"]
    },
    "Output": {
        "MODEL": ["model.svm"]
    }
```

```
}
```

建立模块需要以下 5 步，本文会在后面的章节详细介绍。

- 初始化模块
- 添加输入/输出/参数
- 添加代码实现
- 测试模块

  local 测试

  docker 测试

- 提交模块

8.1.2 安装说明

Screwjack 的使用需要先安装 *docker* http://www.docker.com/ 。

安装 docker

模块的开发环境需要预装 docker，请按照链接自行安装在 Linux 系统中 http://docs.docker.io/installation/ . 安装完毕后，请将当前用户加入到 docker 组中。例如在 Ubuntu Linux 系统中，命令行如下：

```
sudo usermod -aG docker your_linux_username
```

安装 ScrewJack

你可以直接通过 PyPI 直接获取 screwjack :

```
pip install -U screwjack
```

设置 screwjack

在使用 screwjack 前，先建立用户名：

```
export EMR_USERNAME=your_username
```

也可以将用户名设置在 $HOME/.screwjack.cfg 文件中:

```
[user]
username = your_username
```

也可以添加–username 的选项在 screwjack 中：

```
screwjack --username=your_username init
screwjack --username=your_username param_add
screwjack --username=your_username input_add
screwjack --username=your_username output_add
```

8.2 开始使用 SCREWJACK(基本版)

Before you trying following, you should ensure screwjack is installed. Please refer Introduction for detail installation steps.

8.2.1 步骤 1：初始化模块

假设创建一个基本模块，提供非常基本的功能。如果想创建一个“Hive”模块，请参阅文件：使用 Screwjack 创建 Hive 模块。

创建一个基本模块 screwjack :

```
screwjack init basic --name="SVM" --description="A simple SVM"
```

随后会提示设置其他选项，如下所示。在本教程中，我们使用 scikit-learn,它已被封装在基本软件包 zetdata / SCI Python：2.7。

```
Module Version [0.1]:
Module Entry Command [/usr/bin/python main.py]:
```

```
Base Image [zetdata/ubuntu:trusty]: zetdata/sci-python:2.7
Sucessfully created 'svm'
```

也可以使用如下命令行：

```
screwjack init basic --name=SVM --description="A simple SVM" --version="0.1"
--cmd="/usr/bin/python main.py" --base-image="zetdata/sci-python:2.7"
```

你会得到一个基本模块，并被赋予了初始版本号。

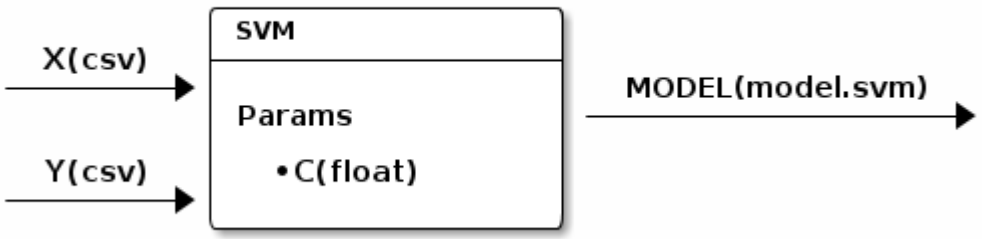
```
svm
|-- Dockerfile
|-- main.py
|-- spec.json
`-- specparser.py

0 directories, 4 files
```

切换到新模块目录，如下步骤是基于这个工作目录进行的。

```
cd svm
```

8.2.2 步骤 2：添加输入/输出/参数

如果创建模块需要两个 输入，一个 输出，和一个参数。如下图所示:



用以下命令添加一个参数：

```
screwjack param_add C
```

用下列命令添加两个输入，第一个参数 X 指的是输入/输出名称，第二个参数 csv 指的是 输入/输出的类型。类型可以是任意用户定义的字符串，如”csv”, “hive.hdfs.table:sub:x“。 关于类型的更多信息，可以参考文件输入输出类型部分。

```
screwjack input_add X csv
screwjack input_add Y csv
```

最后，输出为：

```
screwjack output_add model model.svm
```

8.2.3 步骤 3：实现代码

现在，你可以实现代码如下：

```
vim main.py
```

在本教程中，我们将实现 main.py ：

```
from specparser import get_settings_from_file

from sklearn.svm import LinearSVC
import numpy as np
import pickle

def main():
    settings = get_settings_from_file("spec.json")
    X = np.genfromtxt(settings.Input.X, delimiter=',', skip_header=1)
```

```
    Y = np.genfromtxt(settings.Input.Y, delimiter=',', skip_header=1)
    svc = LinearSVC(C=float(settings.Param.C))
    svc.fit(X,Y)
    with open(settings.Output.MODEL, "w") as f:
        pickle.dump(svc, f)
    print("Done")


if __name__ == "__main__":
    main()
```

如果你想在本模块中添加其他文件，可以直接添加在 Dockerfile 中。

```
vim Dockerfile
```

如果有其它文件添加，可以通过如下命令行加入 Dockerfile ：

```
ADD your_additional_file /home/run/
```

如果需要添加额外目录，可以使用如下命令：

```
ADD your_additional_folder /home/run/your_additional_folder
```

如需更多资料，请参考 Dockerfile 。

8.2.4 步骤 4.1：本地测试

将代码写入模块后，可以通过 screwjack run 进行测试。

```
screwjack run local --help


Usage: screwjack run local [OPTIONS]
Options:
  --param-C TEXT    Param(string)
```

```
--X TEXT            Input
--Y TEXT            Input
--MODEL TEXT        Output
--help              Show this message and exit.
```

我们可以在本地环境进行测试，这个测试环境与实际开发环境比较相似。

```
screwjack run local --param-C=0.1 --X=a.csv --Y=b.csv --MODEL=tmp.model
```

8.2.5 步骤 4.2 在 docker 中进行测试

然后，我们可以尝试在 docker 中执行模块

```
screwjack run docker --param-C=0.1 --X=a.csv --Y=b.csv --MODEL=tmp.model
```

8.2.6 步骤 5：提交模块

你首先提供 spec_server 的 URL 地址用来提交：

```
screwjack submit
```

8.3 使用 SCREWJACK(HIVE)

这里用于说明 screwjack hive 的运行环境。本节将创建一个 EMR（Elastic Map-Reduce）hive 的模块用 来得到搜索查询数据中词频最高的单词及出现的频率。

非常感谢 AOL 的共享。示例数据可以在这里下载，以下为数据格式：

AnonID	Query	QueryTime	ItemRank	ClickURL
142	rentdirect.com	2006-03-01 07:17:12		

142	www.prescriptionfortime.com	2006-03-12 12:31:06		
142	staple.com	2006-03-17 21:19:29		
142	staple.com	2006-03-17 21:19:45		
142	www.newyorklawyersite.com	2006-03-18 08:02:58		
142	www.newyorklawyersite.com	2006-03-18 08:03:09		
142	westchester.gov	2006-03-20 03:55:57	1	http://www.westchestergov.com
142	space.comhttp	2006-03-24 20:51:24		
142	dfdf	2006-03-24 22:23:07		
142	dfdf	2006-03-24 22:23:14		
142	vaniqa.comh	2006-03-		

		25 23:27:12		
142	www.collegeucla.edu	2006-04-03 21:12:14		
142	www.elaorg	2006-04-03 21:25:20		
142	207 ad2d 530	2006-04-08 01:31:04		
142	207 ad2d 530	2006-04-08 01:31:14	1	http://www.courts.state.ny.us
142	broadway.vera.org	2006-04-08 08:38:23		
142	broadway.vera.org	2006-04-08 08:38:31		
142	vera.org	2006-04-08 08:38:42	1	http://www.vera.org
142	broadway.vera.org	2006-04-08 08:39:30		
142	frankmellace.com	2006-04-09		

		02:19:24		
142	ucs.ljx.com	2006-04-09 02:20:44		
142	attornyleslie.com	2006-04-13 00:25:27		
142	merit release appearance	2006-04-22 23:51:18		

8.3.1 步骤 1：初始化 hive 模块

```
screwjack init emr_hive -n hot_token_topN_on_emr -d "Get hotest token in search engine query log."
```

当提示 Module Version 和其他选项时，可以输入回车选择默认选项进行下一步操作。

```
Module Version [0.1]:
Module Entry Command [/usr/bin/python main.py]:
Base Image [zetdata/ubuntu:trusty]:
init emr_hive
Sucessfully created 'hot_token_topn_on_emr'
```

之后，将创建一个名称是 hot_token_topn_on_emr 的目录。

8.3.2 步骤 2：在模块中添加输入/输出和参数

```
screwjack input_add query_log_s3_dir hive.s3.id_query_querytime
```

```
screwjack output_add hot_token_topN_s3_dir hive.s3.table.token_count
screwjack param_add topN string
```

query_log_dir 是 HDFS 中的目录，其中包含原始数据。数据的结构为 ID，查询和查询时间。 hot_token_topN 是 hive 的表名，结果存储在此表里。

8.3.3 步骤 3：（可选）使 UDF 帮助浏览查询到令牌

此步骤为可选项，当模块中需要 UDF 时，参考以下实例说明： example-modules. 创建 UDF 的 jar 包后，将文件放在 ./resource/udfs, HiveRuntime 会自动将文件上传到 AWS 的 S3 中。

8.3.4 步骤 4：编写 Hive 脚本.

可以打开 main.hql 文件，将代码写入：

```
set hive.base.inputformat=org.apache.hadoop.hive.ql.io.HiveInputFormat;

CREATE TEMPORARY FUNCTION splitword AS 'com.your_company.hive.udtf.SplitWord';

--CREATE OUTPUT TABLE

DROP TABLE IF EXISTS hot_token_topN_table;
CREATE EXTERNAL    TABLE hot_token_topN_table
(
        token STRING,
        freq    INT
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE LOCATION '${OUTPUT_hot_token_topN_s3_dir}';
```

```
--CREATE AN EXTERNAL TABLE TO LOAD THE QUERY DATA
DROP TABLE IF EXISTS query;
CREATE EXTERNAL TABLE query
(
        id STRING,
        site STRING,
        timestp TIMESTAMP
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n'
LOCATION '${INPUT_query_log_dir_s3_dir}';

INSERT OVERWRITE TABLE hot_token_topN_table
SELECT token, freq FROM
(
        SELECT token,count(1) AS freq FROM
        (
                SELECT splitword(site) AS token FROM query
        )token_table
        GROUP BY token
)token_frep
ORDER BY freq DESC LIMIT ${PARAM_topN};
```

代码中可以直接引用 screwjack 的输入/输出/参数 ${INPUT_inputname} 和 ${OUTPUT_outputname} ${PARAM_paramname}，在本例为 ${INPUT_query_log_s3_dir}，输出为 ${OUTPUT_hot_token_topN_s3_dir}，参数为 ${PARAM_topN}。

8.3.5 步骤 5：本地测试

测试前，需要将样本数据上传到 S3 存储空间。本例中我们放在了 s3://get-hot-token-kk/input/query 。 模块的输入来自于上一模块的输出，在测试中，需要自己传入输入值。我们使用输入参数文件和输出参数文件来承载输入和输出结果 。本例中输入参数是放在 S3 目录下的一个查询日志文件。创建输入文件./input.param 并存在 s3://get-hot-token-kk/input/query 下。 创建输出文件接收输出结果。

```
screwjack run local
```

然后输入相应的参数来运行测试。

```
Param 'FILE_DIR' [./resources/files]:
Param 'UDF_DIR' [./resources/udfs]:
Param 'AWS_ACCESS_KEY_ID' []: YOUR_AWS_ACCESS_KEY
Param 'AWS_ACCESS_KEY_SECRET' []: YOUR_AWS_ACCESS_KEY_SECRET
Param 'S3_BUCKET' []: get-hot-token-kk
Param 'AWS_Region' []: us-east-1
Param 'EMR_jobFlowId' []: YOUR_EMR_JOB_FLOW_ID
Param 'topN' []: 10
Input 'query_log_s3_dir': input.param
Output 'hot_token_topN_s3_dir': output.param
```

在测试过程中，如果有任何错误或缺陷出现，可以根据日志修改 UDTF 和 hive 脚本。如果调试完毕， 定义好的输出参数将被创建和写入 utput.param 。如果测试成功完成，我们在 S3 目录中可以得到一个 在 output.param 文件。会得到 s3://get-hot-token-kk/zetjob/your_username/job456/blk789/OUTPUT_hot_token_topN_s3_dir. 打开 S3 文件会看到单词频率的排序前十位的结果如下：

```
of,110575
-,104052
in,91521
the,82961
for,70107
and,66675
to,45168
free,45149
a,36220
google,34970
```

8.3.6 步骤 6：在 docker 中进行测试

```
screwjack run docker
```

这一步是测试模块能否在 docker 镜像中运行。起初，screwjack 将在 hive 运行环境下建立一个特定的映像 然后将在 UDFS 映像下测试脚本。 如果运行结果成功，证明将此模块已通过测试，可以上线部署运行。

[1] G. Pass, A. Chowdhury, C. Torgeson, “A Picture of Search” The First International Conference on Scalable Information Systems, Hong Kong, June, 2006.

8.4 模块类型

刚刚开始建立模块会略有难度，根据不同的用户情景我们设计了一套快速建立模块的模板：

Name	Command
Basic	screwjack init basic
Hive(CDH4)	screwjack init hive
PIG(CDH4)	screwjack init pig

EMR(Hive)	screwjack init emr_hive
EMR(Pig)	screwjack init emr_pig

8.5 基础映像

8.5.1 为什么使用基础镜像？

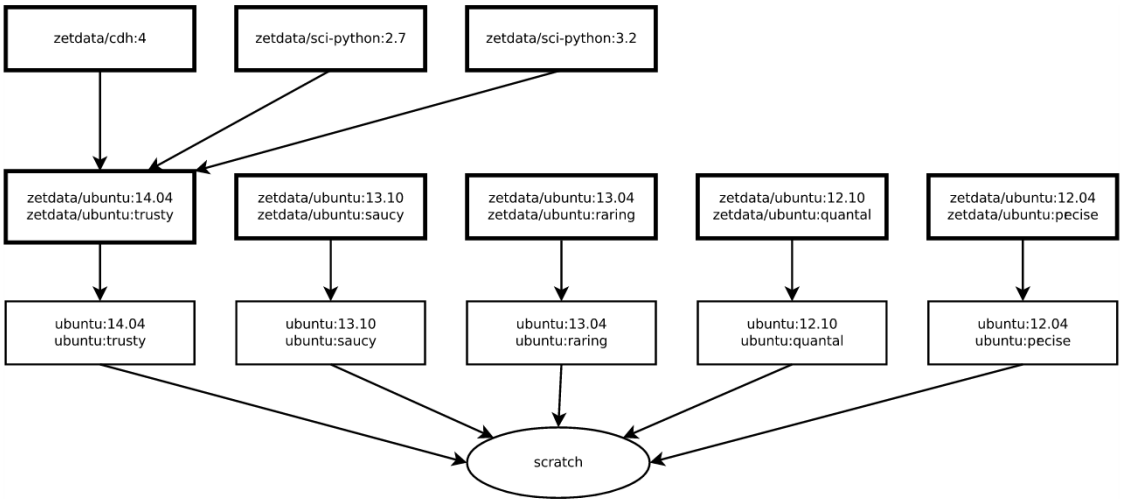
Docker 提供了一个开放的平台用来编译，发布封装的映像。官方提供的 映像可能无法直接满足现有需求，进行模块封装，根据以下原因我们设计自己的基础映像：

1. 基础映像提供了一些最基本的应用，例如 zetdata/cdh:4 是 CDH4 Hadoop cluster 的基础映像。 zetdata/sci-python:2.7 映像提供了 scikit-learn 机器学习工具包。

2. 节省带宽资源。当然，你可以从头开始建立你自己的模块。Docker 支持增量拖拽，可以从远程服务器上进行拖拽，这样就节省你的带宽和时间。

3. 所有编译基础映像的脚本会在 github repo 上开源，映像会推送到 official docker registry 的官方。

8.5.2 Hierarchy of base images



8.6 输入/输出类型

8.6.1 为什么需要类型？

如上图所示，如果两个模块之间需要数据交互，如何是否能将它们连接在一起？模块 A 输出 OA 会写入 csv ， 模块 B 的输入会读取 tsv， ‘module_A’与‘module_B’之间的数据类型不一致会导致任务的失败。

所以我们可以借助 Union Type 设计和规范模块 的输入和输出类型。在 EMR 数据工作流中，前端也提供了输入与输出类型的检查。

一个简单的类型就是一个字符串，例如， “csv”, “csv.salary.table”, “hive.table.tf”。简单类型 是大小写敏感的。两个输入/输出的类型不为空并且类型一致才可以连接。

类型将有助于建立正确的模块间连接。某些错误的使用方式包括：

1. 对很多模块使用相同类型的，这将使类型检查形同虚设。
2. 选择有意义的类型名称。一些类型的名字信息很少，“hive.table.A”，不利于模块的正常使用。