北京郵電大學

Beijing University of Posts and Telecommunications

期末综合

基于中国移动梧桐大数据平台
行业应用实训实验

实验一（三选一）

金融行业“羊毛党”识别案例实践

移动实验一：金融行业“羊毛党”识别案例实践

一、 实验描述

在金融领域，活跃着一批职业“羊毛党”，他们通过套现、套利行为大肆谋利，损害普通用户享有的权益。如何从普通用户中有效鉴别“羊毛党”，从而提前防范，在实际应用中有着重要的意义。

使用大数据技术能从大量普通用户中识别“羊毛党”，商家就可以在设计促销细则时提前规避该群体。假定我们已经有了 1～4 月的潜在用户数据，并知道 2 月的“羊毛党”，可以使用机器学习算法进行数据挖掘，训练出“羊毛党”识别模型，通过 4 月潜在用户数据推理并输出潜在“羊毛党”。

二、 实验目的

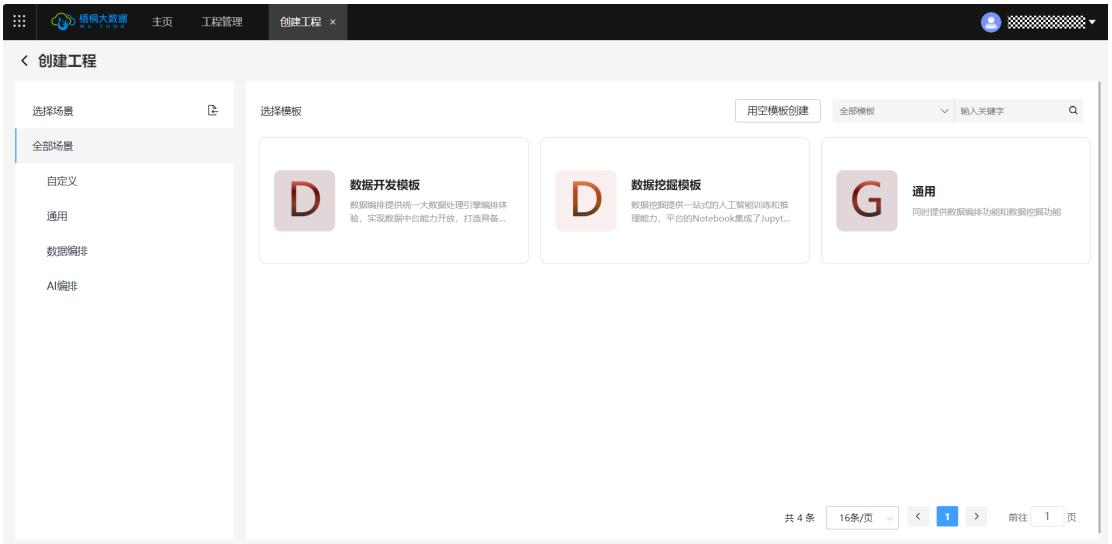
利用移动的梧桐大数据平台，基于机器学习算法进行数据挖掘从而预测“羊毛党”，从而了解分类任务并了解其原理。

三、 实验环境

梧桐大数据实训平台

四、 实验步骤

附：截图要求：能看到右上角用户信息，右下角若有时间需要将时间也截进来。



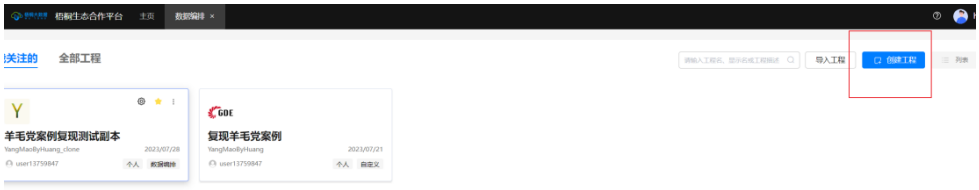
使用梧桐·鸿鹄大数据实训平台的数据编排工具进行数据准备，并通过该平台的数据挖掘工具进行 AI 模型数据挖掘，预测未来两个月的疑似“羊毛党”。

4.1 工程准备

步骤一：

工程创建：

创建工程，工程作为基本管理单元可进行编排开发和数据模型管理。在数据编排工具首页，单击“创建工程“按钮，如下图所示：

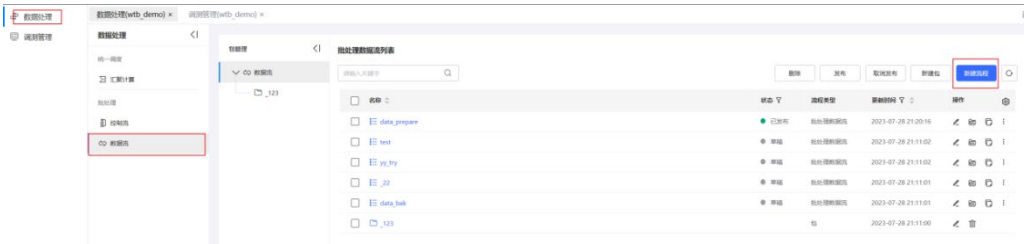


选择通用模板，然后在弹出的“工程信息”对话框中输入工程相关信息，如下图所示。

步骤二：

数据流程编排

可通过图形化界面按钮进 行数据加工。打开步骤一创建的工程，在导航栏单击“数据处理”进入数据处理界面，点击“数据流”，然后在右侧点击“新建流程”，如下图所示，弹出新建数据流对话框，输入名称[名称要求为姓名拼音+学号]即可完成数据流的创建。

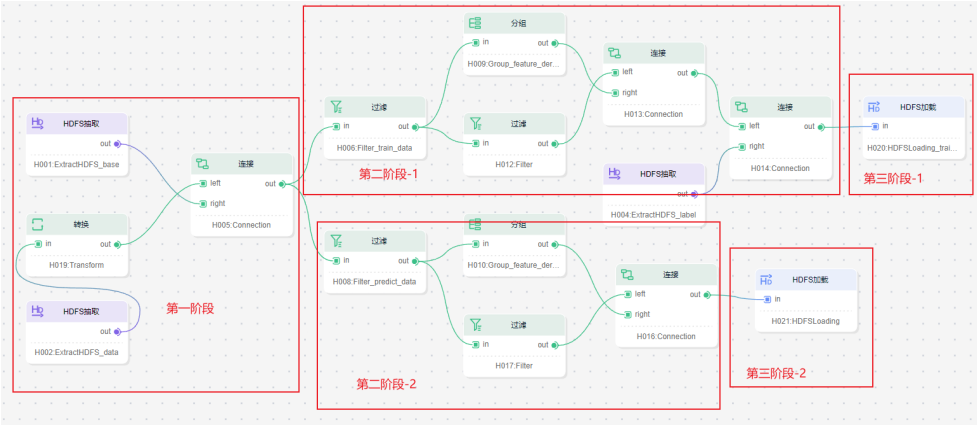


[给出流程创建后的截图，有清晰的流程名称]

步骤三：

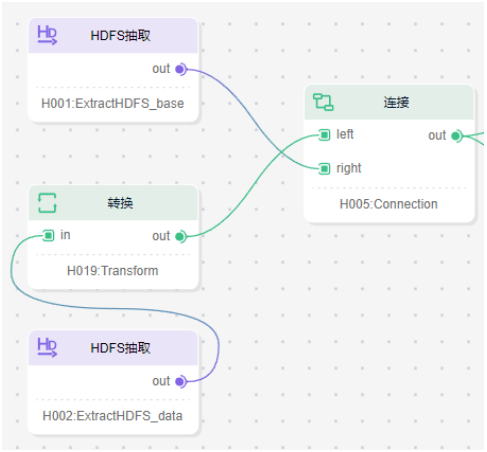
在数据流画布中进行算子的编排

编排包括 3 个阶段：第一阶段是抽取数据，即从 HDFS 中抽取本案例的数据到编排的数据流中；第二阶段进行数据的处理，即根据实际需要进行表关联或字段计算统计等；第三阶段是将处理完成后的数据加载成文件并存放到 HDFS 中。数据流整体算子编排如下图所示。接下来对整个流程进行详细介绍。



4.2 第一阶段算子编排

第一阶段算子编排如下图所示：



（1）HDFS 抽取算子：从 Hadoop 中抽取数据。系统将按照指定的文件位置和文件读取形式进行抽取。

（2）转换算子：用于对输入文件或数据集中的一个字段或多个字段进行表达式计算。

（3）连接算子：用于将两个数据集按字段进行连接。系统将从两个源数据集中按关键字段查找，并根据连接类型输出字段。

（4）读取“用户信息表”和“用户行为表”，并对“用户行为表”的数据进行转换，最终将两张表通过连接算子连接为一张表。

步骤一

数据的 HDFS 路径为/tmp/wutong/example_data，该路径下包括 3 个数据文件，分别是 feature_base.csv（用户信息表）、feature_data.csv（用户行为表）和 train_label.csv（用户标签）。操作配置如下图所示。

H001:ExtractHDFS_base HDFS 抽取算子基础配置如图：

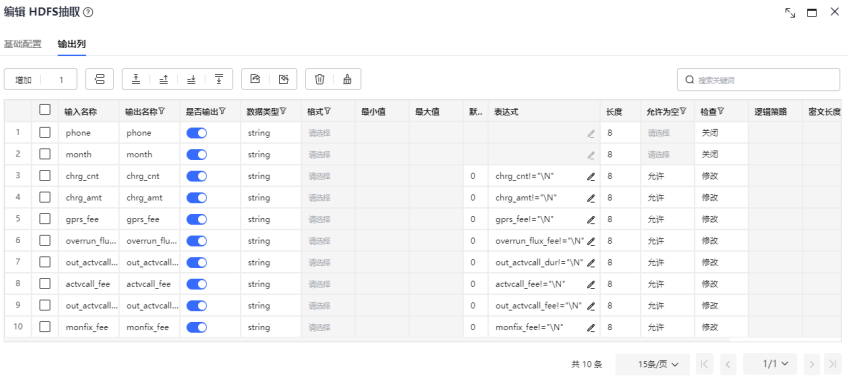
【下述 HDFS 抽取算子的配置页面，不需要先选择物理模型，直接填入文件路径和文件名即可进行数据的抽取】

H001:ExtractHDFS_base HDFS 抽取算子输出列如图：

【请注意：输出列中需要抽取的字段，要手动进行添加】

H002:ExtractHDFS_data HDFS 抽取算子基础配置如图：

H002:ExtractHDFS_data HDFS 抽取算子输出列如图：



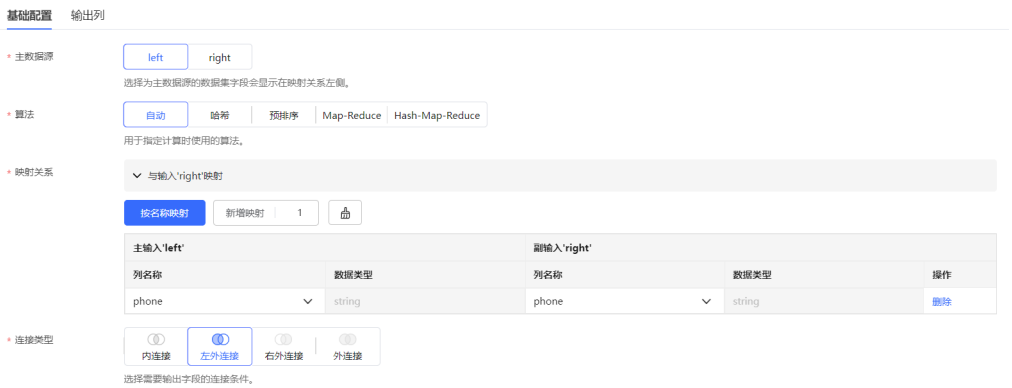
步骤二

在转换操作中 phone 和 month 字段保留原有字段输出，其余所有字段均使用 doubleconvert 函数进行转换，例如填写 chrg_cnt 字段表达式为 doubleconvert(chrg_cnt)，如下图所示。

步骤三

连接时，将转换算子处理后的“用户行为表”作为主数据源，算法选择自动，映射关系选择“按名称映射”，连接类型为左外连接。两个表中都存在 phone 字段，由于使用左外连接，因此不输出右侧表的 phone 字段。

其基础配置如下图：



输出列如下图：

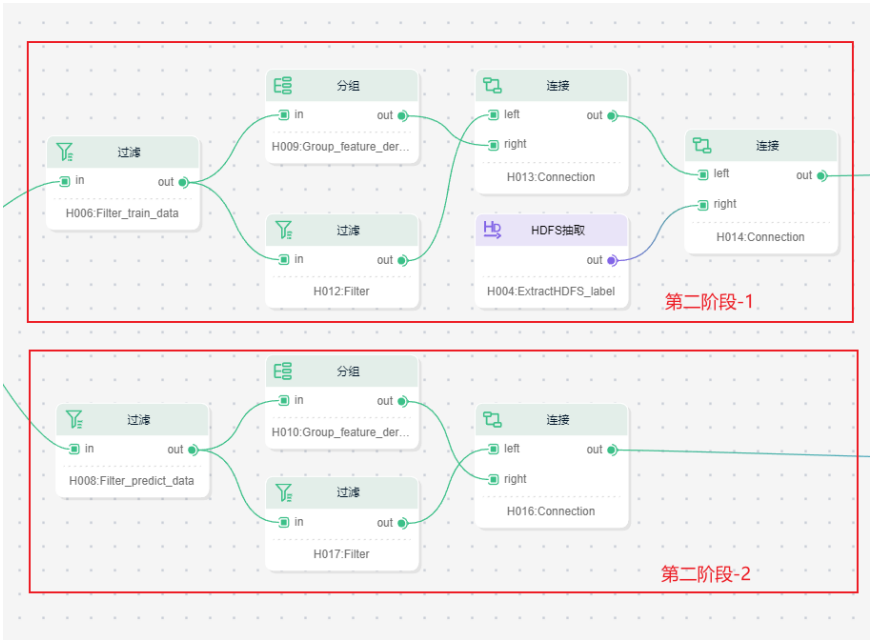
基础配置 输出列

	是否输出	输出名称	组名	输入名称	数据类型	格式
1	是	phone	left	phone	string	
2	是	month	left	month	string	
3	是	chrg_cnt	left	chrg_cnt	double	
4	是	chrg_amt	left	chrg_amt	double	
5	是	gprs_fee	left	gprs_fee	double	
6	是	overrun_flux_fee	left	overrun_flux_fee	double	
7	是	out_actvcall_dur	left	out_actvcall_dur	double	
8	是	actvcall_fee	left	actvcall_fee	double	
9	是	out_activcall_fee	left	out_activcall_fee	double	
10	是	monfix_fee	left	monfix_fee	double	
11	否	phone	right	phone	string	
12	是	if_family	right	if_family	integer	
13	是	if_group	right	if_group	integer	

[给出最终的输出列截图,给出整个电脑屏幕的截图，需要能够看到时间信息]

4.3 第二阶段算子编排

第二阶段算子编排，如下图所示。



该部分有以下几个算子：

（1）过滤算子：相当于关系数据库中 where 条件，用于保留符合条件的数据。

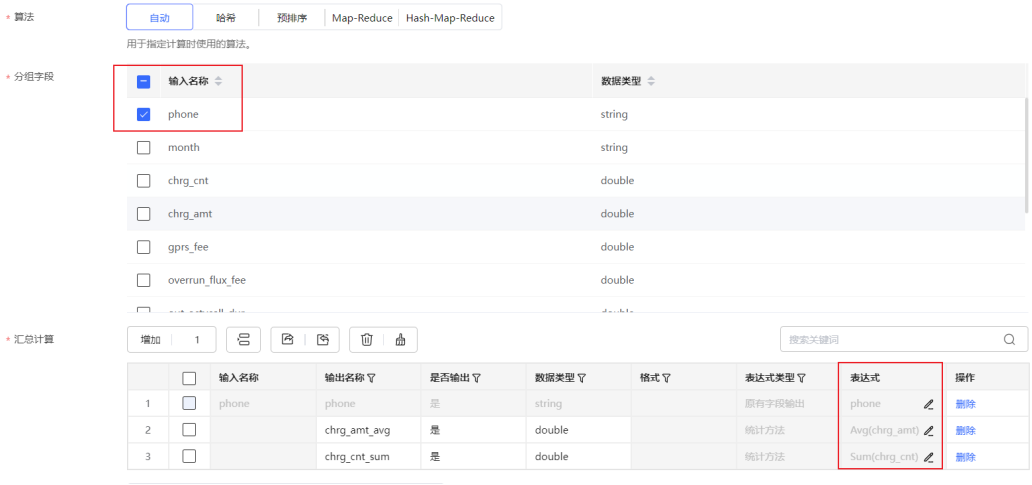
（2）分组算子：相当于关系数据库中的 group by 分组，将表按照指定字段进行分组，同时支持各种汇总计算（包括求平均值、求和、求最大值和求最小值等）。

步骤一：

第二阶段-1 筛选出 2020 年 01 月和 2020 年 02 月的数据；然后一部分按照 phone 进行分组，分别汇总计算出 chrg_amt_avg（使用表达式 Avg(chrg_amt)）和 chrg_cnt_sum（使用表达式 Sum(chrg_cnt)）两个字段；另一部分则单独过滤出 2020 年 02 月的数据；再将这两部分连接为一张表；最终将该表和用户标签连接为一张表作为训练数据。其中筛选出 2020 年 01 月和 2020 年 02 月的表达式分别为 month=='202001'和 month=='202002'，同理可得单独获得 2020 年 02 月数据的表达式。在单独获得 02 月的过滤算子中不输出 month、chrg_cnt 和 chrg_amt 字段，其余字段正常输出。在和用户标签连接时选择右外连接。

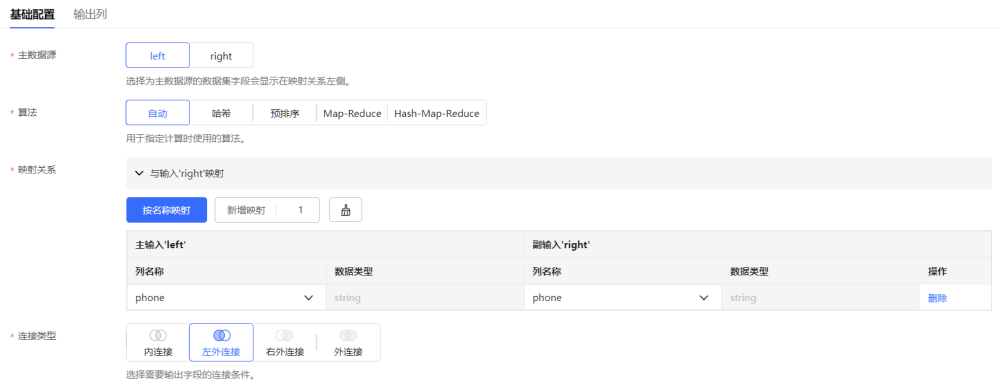
H006: Filter_train_data 过滤算子输出列如下图

H009:Group_feature_derived 分组算子配置如下图



H012: Filter 过滤算子输出列如下图

H013: Connection 连接算子基础配置如下图



H013: Connection 连接算子输出列如下图

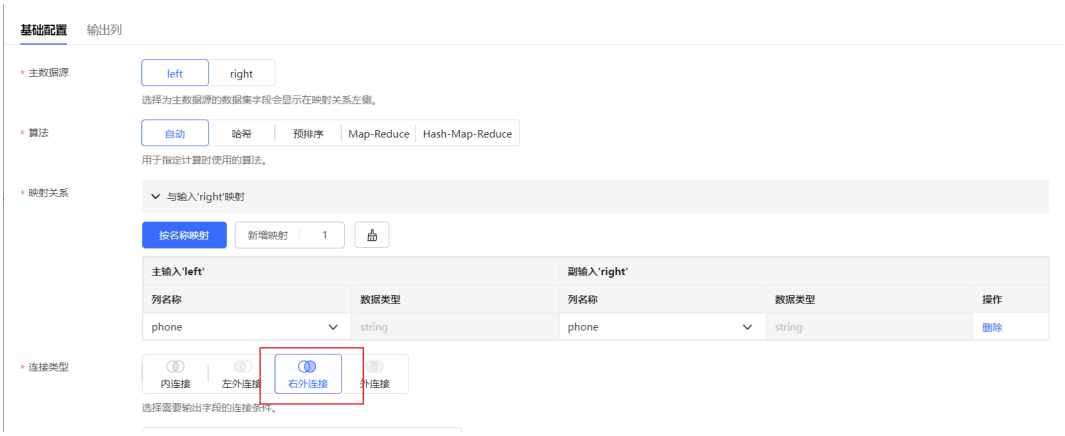
	是否输出	输出名称	组名	输入名称	数据类型	格式
1	是	phone	left	phone	string	
2	是	gprs_fee	left	gprs_fee	double	
3	是	overrun_flux_fee	left	overrun_flux_fee	double	
4	是	out_actvcall_dur	left	out_actvcall_dur	double	
5	是	actvcall_fee	left	actvcall_fee	double	
6	是	out_activcall_fee	left	out_activcall_fee	double	
7	是	monfix_fee	left	monfix_fee	double	
8	是	if_family	left	if_family	integer	
9	是	if_group	left	if_group	integer	
10	否	phone	right	phone	string	
11	是	chrg_amt_avg	right	chrg_amt_avg	double	
12	是	chrg_cnt_sum	right	chrg_cnt_sum	double	

H004: ExtractHDFS_label HDFS 抽取算子基础配置如下图

H004: ExtractHDFS_label HDFS 抽取算子输出列如下图

		输入名...	输出名...	是否输出	数据类...	格式	最小值	最大值	默认值	表达式	长度	允许为...	检查	描述	操作
1		phone	phone	是	string						8		关闭	请输入	删除
2		label	label	是	double						8		关闭	请输入	删除

H014: Connection 连接算子基础配置如下图



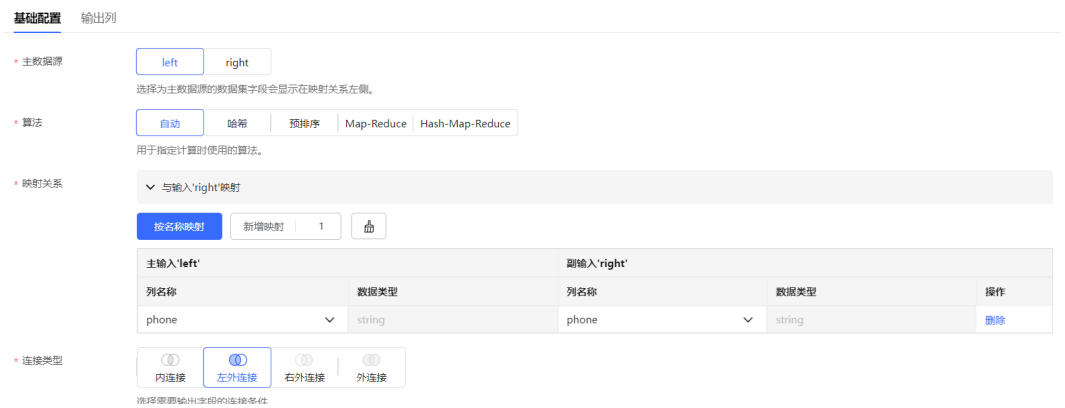
H014: Connection 连接算子输出列如下图

	是否输出	输出名称	组名	输入名称	数据类型	格式
1	是	phone	left	phone	string	
2	是	gprs_fee	left	gprs_fee	double	
3	是	overrun_flux_fee	left	overrun_flux_fee	double	
4	是	out_actvcall_dur	left	out_actvcall_dur	double	
5	是	actvcall_fee	left	actvcall_fee	double	
6	是	out_activcall_fee	left	out_activcall_fee	double	
7	是	monfix_fee	left	monfix_fee	double	
8	是	if_family	left	if_family	integer	
9	是	if_group	left	if_group	integer	
10	是	chrg_amt_avg	left	chrg_amt_avg	double	
11	是	chrg_cnt_sum	left	chrg_cnt_sum	double	
12	否	phone	right	phone	string	
13	是	label	right	label	double	

步骤二：

第二阶段-2 筛选出 2020 年 03 月和 2020 年 04 月的数据；一部分按照 phone 分组，分别汇总计算出 chrg_amt_avg 和 chrg_cnt_sum 字段；另一部分单独过滤出 2020 年 04 月的数据；再将两部分连接为一张表作为预测数据。配置和不输出字段和步骤一的相同，在单独获得 04 月数据的过滤算子中不输出 month、chrg_cnt 和 chrg_amt 字段。两部分连接为一个表使用的左外连接。

H016: Connection 连接算子基础配置如下图



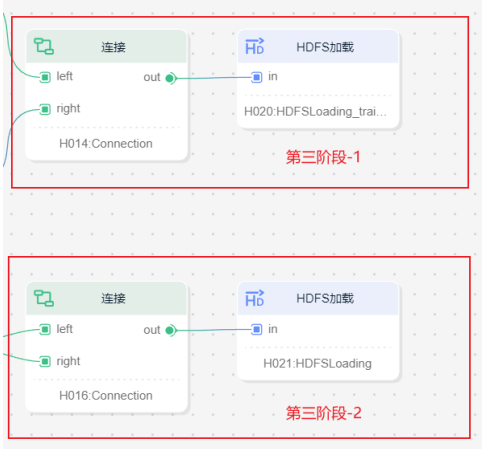
H016: Connection 连接算子输出列如下图

	是否输出	输出名称	组名	输入名称	数据类型	格式
1	是	phone	left	phone	string	
2	是	gprs_fee	left	gprs_fee	double	
3	是	overrun_flux_fee	left	overrun_flux_fee	double	
4	是	out_actvcall_dur	left	out_actvcall_dur	double	
5	是	actvcall_fee	left	actvcall_fee	double	
6	是	out_activcall_fee	left	out_activcall_fee	double	
7	是	monfix_fee	left	monfix_fee	double	
8	是	if_family	left	if_family	integer	
9	是	if_group	left	if_group	integer	
10	否	phone	right	phone	string	
11	是	chrg_amt_avg	right	chrg_amt_avg	double	
12	是	chrg_cnt_sum	right	chrg_cnt_sum	double	

[给出连接算子输出列截图，给出整个电脑屏幕的截图，需要能够看到时间信息]

4.4 第三阶段算子编排

第三阶段算子编排，如下图所示。



（1）HDFS 加载算子：类似写文件，将处理后的数据加载到 Hadoop 集群中。

（2）将训练数据集和预测数据集分别保存为 CSV 文件并加载到 Hadoop 中,加载到自己活动的编排输出路径下。其中命名方式分别为名字全拼_学号_train_data.csv 和名字全拼_学号_predict_data.csv，例如：张三的训练数据集为/srv/multi-tenant/midteant02/dev/user/px_bupt/px_bupt_xxx/etl_output/zhangsan_2024xxxx_train_data.csv

操作配置如下。

HDFS 加载算子配置如下图

训练数据集输出列如下图

		源字段	目标字段	源字段序号	是否输出	数据类型	格式	对齐方式	精度	变量	描述
1		phone	phone	1	是	string		left	0		请输入
2		gprs_fee	gprs_fee	2	是	double		left	0		请输入
3		overrun_flux...	overrun_flux...	3	是	double		left	0		请输入
4		out_actvcall_...	out_actvcall_...	4	是	double		left	0		请输入
5		actvcall_fee	actvcall_fee	5	是	double		left	0		请输入
6		out_activcall...	out_activcall...	6	是	double		left	0		请输入
7		monfix_fee	monfix_fee	7	是	double		left	0		请输入
8		if_family	if_family	8	是	integer		left	0		请输入
9		if_group	if_group	9	是	integer		left	0		请输入
10		chrg_amt_avg	chrg_amt_avg	10	是	double		left	0		请输入
11		chrg_cnt_sum	chrg_cnt_sum	11	是	double		left	0		请输入
12		label	label	12	是	double		left	0		请输入

预测数据集输出列如下图

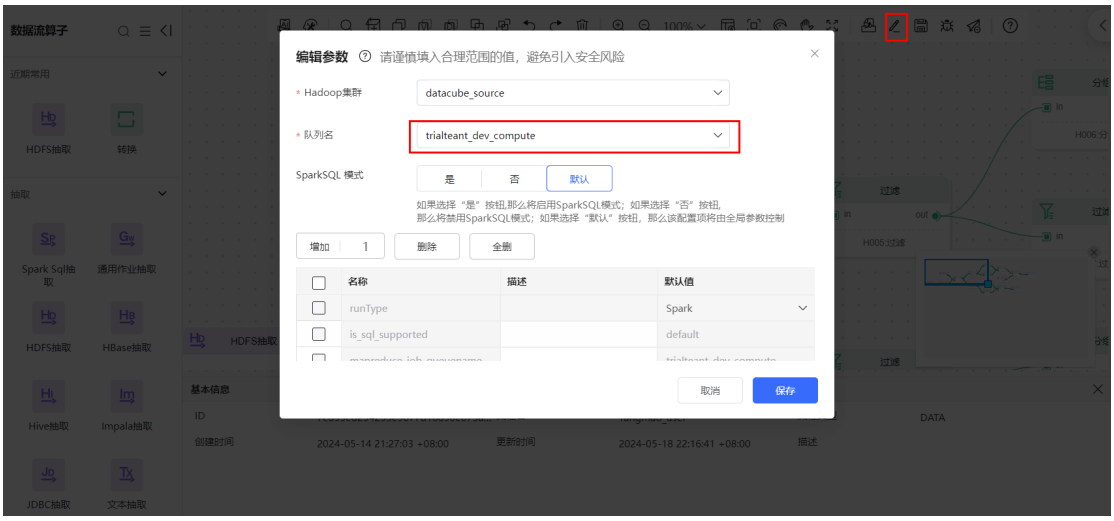
		源字段	目标字段	源字段序号	是否输出	数据类型	格式	对齐方式	精度	变量	描述
1		phone	phone	1	是	string		left	0		请输入
2		gprs_fee	gprs_fee	2	是	double		left	0		请输入
3		overrun_flux...	overrun_flux...	3	是	double		left	0		请输入
4		out_actvcall_...	out_actvcall_...	4	是	double		left	0		请输入
5		actvcall_fee	actvcall_fee	5	是	double		left	0		请输入
6		out_activcall...	out_activcall...	6	是	double		left	0		请输入
7		monfix_fee	monfix_fee	7	是	double		left	0		请输入
8		if_family	if_family	8	是	integer		left	0		请输入
9		if_group	if_group	9	是	integer		left	0		请输入
10		chrg_amt_avg	chrg_amt_avg	10	是	double		left	0		请输入
11		chrg_cnt_sum	chrg_cnt_sum	11	是	double		left	0		请输入

[给出算子配置截图，要求能够看到文件名，给出预测数据输出列，要求整个电脑屏幕截图，能看到时间信息]

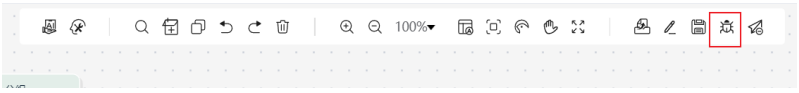
4.5. 在线执行数据流

步骤一：

1、点击编辑参数按钮，选择集群及队列资源



2、在线调测数据流，执行数据流程处理，可根据实际需要进行分支调测或算子断点调测，如下图所示。



步骤二：

可以挂起分支，如下图所示。

步骤三：

支持对某些算子进行断点调测，如下图所示。

支持在整个数据流程中的某些算子打上断点进行debug。

步骤四：

项目运行流程：环境准备→组件激活→任务提交→任务结束→流程结束。本项目的数据流大致运行时间为十几分钟。运行日志时可同步查看，下图所示。

日志—环境准备如下图：

2023-07-28 21:20:47.23 EXPORTING

2023-07-28 21:20:49.938 MODELING

2023-07-28 21:20:50.0 COMPILING

2023-07-28 21:20:50.426 COMPILED

2023-07-28 21:20:51.891 ACTIVATING

2023-07-28 21:20:51.966 STARTING

2023-07-28 21:20:52.34 RUNNING

日志—各个组件激活如下图：

[2023-07-28 21:20:53,360]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view]flowvars: Get HadoopUser from rscs success. HadoopClusterID:DATACUBE_HADOOP_DS_1, UserName:wutonguser
[2023-07-28 21:20:53,360]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view]flowvars: Get QueueName from rscs success. HadoopClusterID:DATACUBE_HADOOP_DS_1, QueueName:default,manas_data_tenant
[2023-07-28 21:20:53,423]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view]end to getTenantQueueName queueName is default
[2023-07-28 21:20:53,425]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view] H002out before run is start
[2023-07-28 21:20:53,426]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view] H004out before run is start
[2023-07-28 21:20:53,426]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view] H021 before run is start
[2023-07-28 21:20:53,426]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view] H021out before run is start
[2023-07-28 21:20:53,426]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view] H020 before run is start

Spark 任务提交成功如下图：

[2023-07-28 21:20:56,452]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956| [view]thread pool shutdown
[2023-07-28 21:21:03,761]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view]Succeed to submit spark application which appId is: application_1660069463824_2867
[2023-07-28 21:21:07,831]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view]YarnApplicationState is: ACCEPTED
[2023-07-28 21:21:11,836]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view]YarnApplicationState is: ACCEPTED
[2023-07-28 21:21:15,842]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view]YarnApplicationState is: RUNNING

数据流任务全部完成如下图：

[2023-07-28 21:24:42,818]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view]YarnApplicationState is: RUNNING
[2023-07-28 21:24:46,823]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view]YarnApplicationState is: RUNNING
[2023-07-28 21:24:50,832]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view]The monitor in local has finished reading status infomation from HDFS.
[2023-07-28 21:24:50,833]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956|[view]flow state from hdfs is COMPLETED
[2023-07-28 21:24:51,836]||INFO|BPMPool-1-Engine Thread Pool-HDIProcessEngine-526|BDI|PI_28956| [view]thread pool shutdown

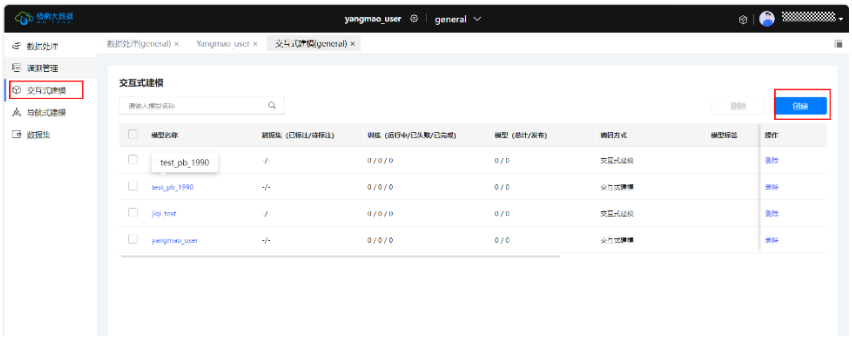
[要求给出数据流截图以及日志截图]

4.6. 数据挖掘

说明：由于本案例的数据已经存储在平台的 HDFS 上，可直接参考步骤三使用 Notebook 进行数据挖掘，可跳过步骤一和步骤二。

步骤一：

使用通用模板创建工程后，点击交互式建模



步骤二：

创建数据集，支持上传数据、进行数据预览和管理，如图所示。

步骤三：

基于数据挖掘工具 Notebook 执行数据挖掘全过程，Notebook 支持交互式开发。在 Notebook 页面中分别实现以下过程。

① 导入接口包

导入所需接口包，包括数据读取、数据处理、模型训练评估所需 Python 库，代码如下。

```
import pandas as pd  #用于处理高级数据结构和数据分析
from manas.dataset import mfile  #用于从 HDFS 上读取数据文件
from sklearn.metrics import roc_auc_score,classification_report #用于评估模型预测结果
from sklearn.model_selection import train_test_split  #用于划分训练数据集和测试数据集
from xgboost import XGBClassifier  #用于构建基于 XGBoost 算法的分类模型
```

其中，pandas 是 Python 的扩展程序库，主要数据结构是 Series（一维数据）与 DataFrame（二维数据），可以对各种数据进行运算操作，比如归并、再成形、选择、数据清洗和数据

加工等，广泛应用在学术、金融、统计学等各个数据分析领域，相关案例详见其官网。

sklearn 是基于 Python 的机器学习工具包，涵盖分类、回归、聚类、降维、模型选择、数据预处理六大模块，降低机器学习实践门槛，将复杂的数学计算集成为简单的函数，并提供众多公开数据集和学习案例，详见其官网。sklearn 的 model-selection 模块提供 train_test_spilt 函数，能够对数据集进行拆分；metrics 模块提供 roc_auc_score、classification_report 等函数，能够对模型输出结果进行评估，如计算分类结果的 AUC、生成分类模型评估报告（包含准确率、召回率、F1 分数）。

此外，为在 Python 环境下应用 XGBoost 算法，需要引入 XGBoost 官方提供的 Python 库，并使用 XGBClassifier 函数构建分类模型，使用细节见其官方文档。

② 读取训练数据集和预测数据集

我们需要借助 mfile.mfile 和 pd.read_csv 函数从 HDFS 中读取所需数据到内存并以 DataFrame 格式进行组织，以便后续的数据操作及模型训练、预测、评估，注意此处将文件地址更改为数据编排阶段实际 HDFS 存储地址，代码如下。

```
train_path = "hdfs://cxq-ns1/tmp/wutong/example_data/train_data.csv" #训练数据集文件地址
file_r = mfile.mfile(train_path, "r")  #获取文件操作
train_df = pd.read_csv(file_r, sep='\,', header=0) #使用 pandas 将数据读取成 DataFrame，完成后直接使用变量操作文件
file_r.close()  #读取完成后关闭数据流
predict_path = "hdfs://cxq-ns1/tmp/wutong/example_data/predict_data.csv" #预测数据集文件地址
file_r = mfile.mfile(predict_path, "r")  #获取文件操作
predict_df = pd.read_csv(file_r, sep='\,', header=0) #使用 pandas 将数据读取成 DataFrame，完成后直接使用变量操作文件
file_r.close()  #读取完成后关闭数据流
```

③ 特征工程处理

特征工程处理包括填充缺失值、导出标签数据、删除训练数据标签和设置数据表索引，代码如下。

```
train_df.fillna(0,inplace = True)                #缺失值填充为 0
predict_df.fillna(0,inplace = True)              #缺失值填充为 0
target_data = train_df[['phone','label']]        #导出标签数据
del train_df['label']                                  #删除训练数据标签
train_data = train_df
train_data = train_data.set_index('phone')       #将 phone 作为 DataFrame 索引
target_data = target_data.set_index('phone')     #将 phone 作为 DataFrame 索引
predict_df = predict_df.set_index('phone')       #将 phone 作为 DataFrame 索引
```

④ 模型训练

配置一定的参数实例化一个基于 XGBoost 算法的二分类模型，以 7 : 3 的比例随机划分训练数据集和测试数据集并使用训练数据集完成模型的训练，针对测试数据集输出预测结果并计算 AUC 进行模型训练效果的初步评估，代码如下。

```
model = XGBClassifier(min_chile_weight = 1,max_depth = 10,learning_rate = 0.05,gamma = 0.4,colsample_bytree = 0.4)  #构建 XGBoost 二分类模型
x_train,x_test,y_train,y_test = train_test_split(train_data,target_data,test_size = 0.3,random_state = 42)  #随机划分训练数据集和测试数据集
model.fit(x_train,y_train)  #训练模型
predict_l = model.predict_proba(x_test)[:,1]  #输出预测结果
auc = roc_auc_score(y_test,predict_l)  #计算 AUC
print('AUC is {}'.format(auc))  #输出 AUC
```

上述代码使用的重要类及函数说明如下。

a．xgboost.XGBClassifier 类：XGBoost 分类器的 sklearn API 实现。

该类部分参数说明如下表所示。

参数	说明
min_chile_weight	最小叶子节点样本权重和，如果决策树分裂产生的叶子节点上样本权重之和小于该值，就停止分裂
max_depth	基础学习器的决策树的最大深度
learning_rate	学习率
gamma	决定一个叶子节点是否应该进一步分割的阈值，如果损失函数的差值（通常称为增益）在潜在分裂后小于该值，则不执行分裂
colsample_bytree	构造每棵决策树时特征的子采样率

fit 方法用于训练分类模型，根据参数中的特征矩阵 X 和标签 y 进行分类模型的训练，并返回一个训练好的分类模型。

predict_proba 方法用于预测结果，根据参数中的特征矩阵 X 预测每一个测试样本是给定类别的概率，并返回所有样本被分类为给定类别的概率。

b．sklearn.model_selection.train_test_split 函数：将数组或矩阵划分为训练数据集和测试数据集。

该函数部分参数说明如下表所示。

参数	说明
*arrays	待划分的数据
test_size	如果是 0～1 的浮点数，则为测试数据集的所占比例
random_state	随机状态，控制划分前的数据混洗

该函数以列表的形式返回划分后的训练数据集和测试数据集。

c．sklearn.metrics.roc_auc_score 函数：根据预测结果计算 AUC。

该函数部分参数说明如下表所示。

参数	说明
y_true	真实标签
y_score	二分类时为较大标签的预测概率

该函数以浮点数的形式返回 AUC。

⑤ 特征重要度分析

完成模型训练后可以输出模型的特征重要度，分析各个特征对模型结果的影响。本案例将模型特征及其重要度组织成 DataFrame 格式并按照特征重要度降序输出，代码如下。

```
pd.DataFrame({'feature':x_train.columns,'importance':model.feature_importances_}).sort_values(
ascending = False,by = 'importance')  #降序输出模型特征重要度
```

代码中使用到 xgboost.XGBClassifier 类的 feature_importances_属性，其根据一定的评估方法（如总增益）计算各个特征重要度。

模型特征重要度如下图所示。

	模型特征	重要度
7	monfix_fee	0.363914
2	gprs_fee	0.228949
4	out_actvcall_dur	0.076633
5	actvcall_fee	0.058394
0	if_family	0.038820
3	overrun_flux_fee	0.037869
11	down_flux	0.031884
9	call_cnt	0.027817
6	out_activcall_fee	0.027171
10	up_flux	0.018737
16	chrg_cnt_sum	0.018350
13	p2psms_up_cnt	0.017496
17	chrg_amt_avg	0.016305
8	gift_acct_amt	0.014850
14	p2psms_cmnct_fee	0.013631
12	sms_inpkg_ind	0.003493
15	p2psms_pkg_fee	0.003385
1	if_group	0.002301

⑥ 定制模型性能指标评估报告

为选择划分正常用户和“羊毛党”的最佳概率阈值，首先定义模型性能指标评估报告函数，对不同概率取值进行“羊毛党”的划分，然后根据真实标签和预测标签计算精确率、召回率和 F1 分数，最后汇总所有概率取值的评估结果，生成评估报告，代码如下。

```
#模型性能指标评估报告函数
def get_threshold_report(y_predict, target_name):
    model_count = y_predict[target_name].value_counts().sort_index()[1]  #获得测试数据集中"羊毛党"的用户数量
    y_test = y_predict[[target_name]]  #单独取出标签列
    model_test = y_predict.copy()
    report = pd.DataFrame()  #初始化 report 为 DataFrame 格式
    for i in range(1, 20):  #等间隔选取 0.05～0.95 的 20 个不同概率阈值
        sep_pr = []  #记录本次划分结果
        sep_value = i * 0.05  #本次阈值
        col_name = 'sep_.' + str(round(sep_value, 2))  #本次列名，用于记录预测标签
```

```
        model_test[col_name] = model_test['predict'].apply(lambda x: 1 if x > sep_value else 0)  #根据预测概率及阈值生成预测标签
        sep_pr.append(str(round(sep_value, 2)))  #记录本次阈值
        sep_pr.append(model_count)  #记录真实"羊毛党"的用户数量
        predict_model = model_test[col_name].value_counts().sort_index()
#获取预测标签分类统计数量
        if predict_model.shape[0] == 1:  #只有一类标签的情况
            if predict_model.index.format() == '0':  #一类标签为 0，即不存在被预测为"羊毛党"的用户
                sep_pr.append(0)  #记录被预测为"羊毛党"的用户数量为 0
            else:
                sep_pr.append(predict_model[0])  #一类标签为 1，即所有用户均被预测为"羊毛党"，记录被预测为"羊毛党"的用户数量，即 predict_model 的第一行数据
        else:  #两类标签的情况
            sep_pr.append(predict_model[1])  #记录被预测为"羊毛党"的用户数量，即 predict_model 的第二行数据
        model_report = classification_report(y_test, model_test[col_name].values, digits=4)  #根据真实标签和预测标签生成分类报告
        pr_str = ' '.join(model_report.split('\n')[3].split()).split(' ')
#提取分类报告最后一行的三种评估指标
        sep_pr.append(pr_str[1])  #记录精确率
        sep_pr.append(pr_str[2])  #记录召回率
        sep_pr.append(pr_str[3])  #记录 F1 分数
        report = pd.concat([report, pd.DataFrame([sep_pr])], axis=0)  #拼接本次划分的评估报告
    report.columns = ['threshold', 'actual', 'predict', 'precision', 'recall', 'f1-score']  #设置评估报告列名
    report = report.reset_index(drop=True)  #重置索引，并将原索引删除
    return report  #返回模型性能指标评估报告
```

上述代码使用的重要类及函数说明如下。

sklearn.metrics.classification_report 函数：生成主要分类评估指标的文本报告。

该函数的部分参数说明如下表所示。

参数	说明
y_true	真实标签
y_pred	分类模型的预测标签
digits	输出浮点数的保留位数

该函数以文本形式返回每一类的精确率、召回率和 F1 分数。

⑦ 确定最佳划分概率阈值

利用定义好的模型性能指标评估报告函数 get_threshold_report 生成如图 2-6-41 所示的性能指标评估报告，可以观察到最大 F1 分数是 0.8109 且对应的概率阈值为 0.3，因此我们选

择该概率阈值作为“羊毛党”的划分依据，代码如下。

```
y_test['predict'] = predict_1  #添加预测结果列
get_threshold_report(y_test, 'label')  #生成性能指标评估报告
```

性能指标评估报告如下图：

	threshold	actual	predict	precision	recall	f1-score
0	0.05	1806	3024	0.5956	0.9972	0.7458
1	0.1	1806	2813	0.6374	0.9928	0.7764
2	0.15	1806	2682	0.6655	0.9884	0.7955
3	0.2	1806	2628	0.6758	0.9834	0.8011
4	0.25	1806	2453	0.7036	0.9557	0.8105
5	0.3	1806	2451	0.7042	0.9557	0.8109
6	0.35	1806	2443	0.7036	0.9518	0.8091
7	0.4	1806	2409	0.7069	0.9430	0.8081
8	0.45	1806	1491	0.7746	0.6395	0.7006
9	0.5	1806	1490	0.7745	0.6390	0.7002
10	0.55	1806	1289	0.7719	0.5509	0.6430
11	0.6	1806	85	0.9765	0.0460	0.0878
12	0.65	1806	85	0.9765	0.0460	0.0878
13	0.7	1806	85	0.9765	0.0460	0.0878
14	0.75	1806	130024	0.0000	0.0000	0.0000
15	0.8	1806	130024	0.0000	0.0000	0.0000
16	0.85	1806	130024	0.0000	0.0000	0.0000
17	0.9	1806	130024	0.0000	0.0000	0.0000
18	0.95	1806	130024	0.0000	0.0000	0.0000

⑧ 模型推理

使用训练好的二分类模型对测试数据集的数据进行预测，得到每个用户疑似“羊毛党”的概率并根据选定的概率阈值（0.3）划分“羊毛党”，即将预测概率大于 0.3 的用户划分为“羊毛党”，保存并输出预测结果。模型推理结果如下图所示，代码如下。

phone	predict_proba	predict_label
f2e79de2c2d8c7cfe1ade5c5353d4b30	0.718698	1
914ae39e814c0ba946b5e3589f9bdff7	0.590268	1
51b0f2b2d5893d8289c50131d0e474c5	0.590268	1
a9badf48fef5c853c744bd4b895d34ac	0.590268	1
c28a310d76fa3adf57fe4a8e8fc69f1d	0.560068	1
...	...	...
631239806e58f767c23c36c2fa2ef7f4	0.003229	0
9ca2b1886ac1fa1c5e9086ba8d15e920	0.003229	0
02b97084fe686c19317a5a0b1b400bed	0.003229	0
e22117f6121501c8457ae92bd5d71ed9	0.003228	0
3d892979c18008675d401e1373ff21ad	0.003228	0

190143 rows × 2 columns

```
#输出模型推理结果
output = predict_df.copy()  #获取待预测数据
```

```
output['predict_proba'] = model.predict_proba(output)[:, 1]  #使用模型进行预测，得到预测概率
output['predict_label'] = output['predict_proba'].apply(lambda x: 1 if x > 0.3 else 0)  #根据选定的概率阈值划分"羊毛党"
df_output = output[['predict_proba', 'predict_label']].sort_values(ascending=
False, by='predict_proba')  #按照预测概率降序排列疑似"羊毛党"
df_output.to_csv('output.csv')  #保存预测结果
df_output  #输出预测结果
```

[给出 notebook 代码截图、性能指标评估报告以及预测结果截图]

⑨ （可选）配置训练作业

如果需要离线进行模型训练和模型推理的调度执行，可以在系统上配置训练作业，如下图所示。

五、 实验结果与分析

1、 提交相关截图：（具体截图要求见指导书内容，每个截图 1 分）

（1）流程创建完成截图

（2）第一阶段算子编排最终输出列截图

（3）第二阶段算子编排 Connection 连接算子输出列截图

（4）第三阶段算子编排算子配置截图

（5）第三阶段算子编排预测数据输出列截图

（6）数据流截图

（7）日志截图

（8）性能指标评估报告

（9）预测结果截图

（10）notebook 代码截图

2、 相关文字描述（文字描述 5 分）

3、 代码分析以及实验的总结（可以包括实验中遇到的问题、自己的思考等，5 分）