机器学习-电影推荐

本项目展示了如何处理一个包含用户对不同电影的评分的数据集，针对数据集中的用户对某电影评分进行预测，并对于用户可能感兴趣的电影进行推荐。

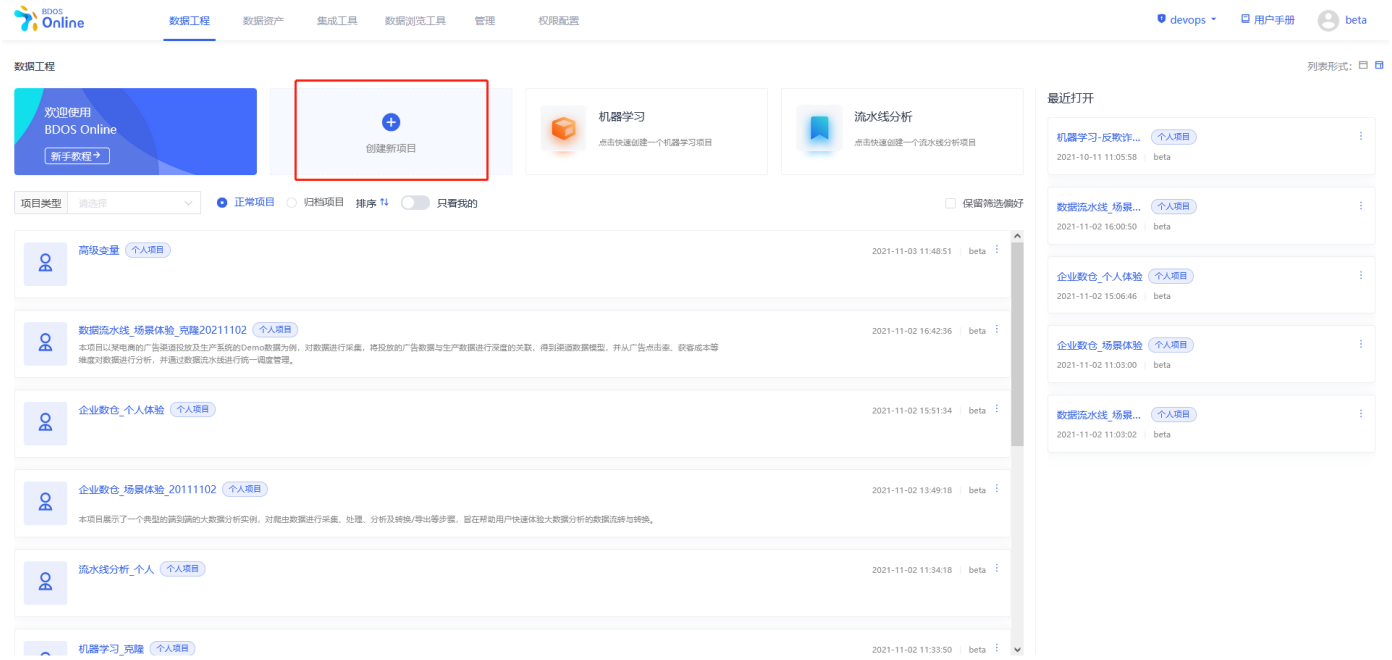
数据集数据来源：GroupLens 官方网站，数据集来源：movielens_link。文件包含2017年7月或之前发行的电影数据4.5万条，包含来自27万位用户对于4.5万部电影的2600万个评分，评分范围为0-5。

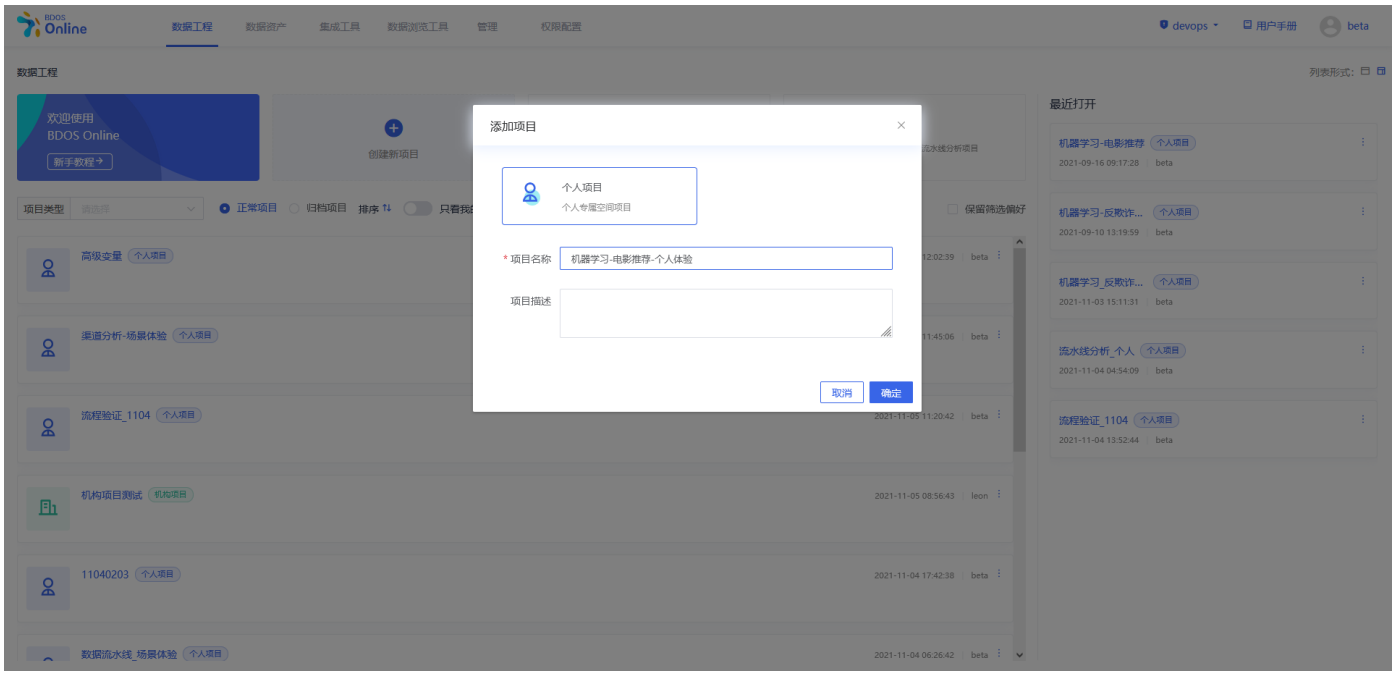
备注：因此Demo项目设计方案的调整，录屏列表视图中出现的HDFS到Hive步骤已删除，但不影响此Demo的其他步骤及其运行结果。

本介绍以视频的形式，向大家展示一个机器学习通过ALS模型进行电影个性化推荐的实例，过程包括数据采集、数据处理和数据分析这几个步骤：

步骤一：新建个人/机构项目

用户点击界面上创建新项目，填写名称可参照下图、



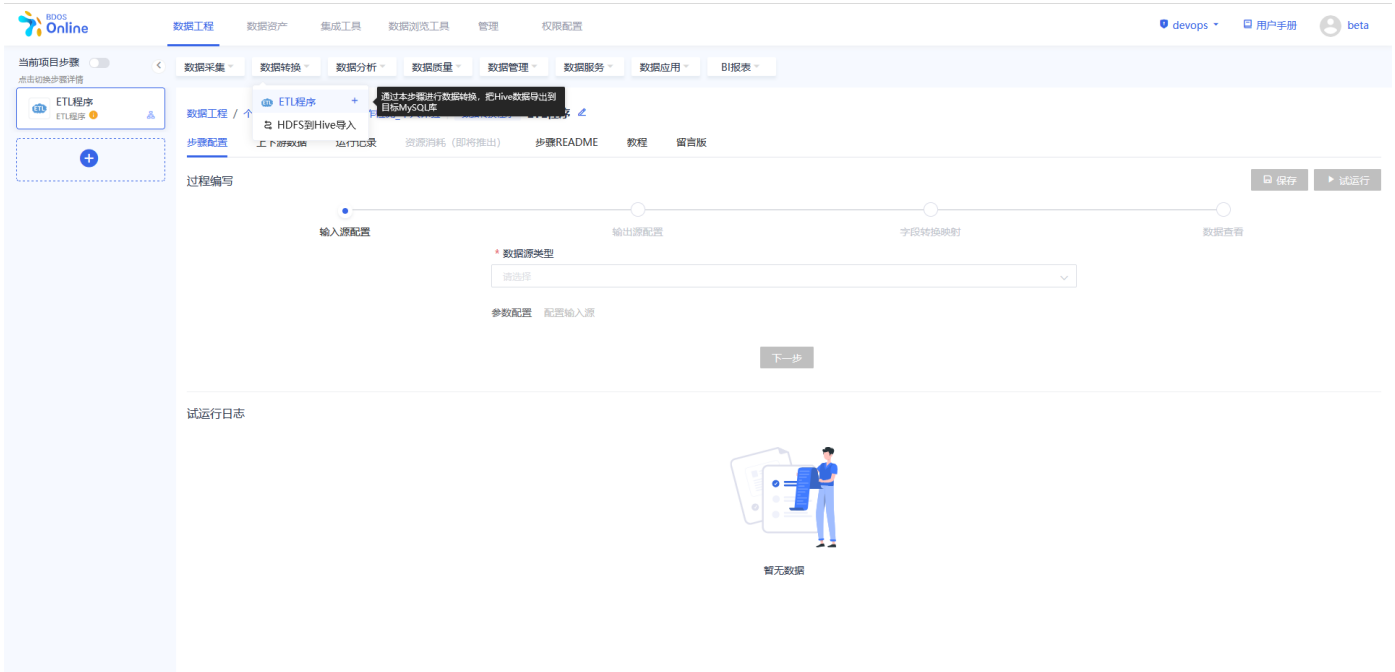


步骤二：添加项目步骤

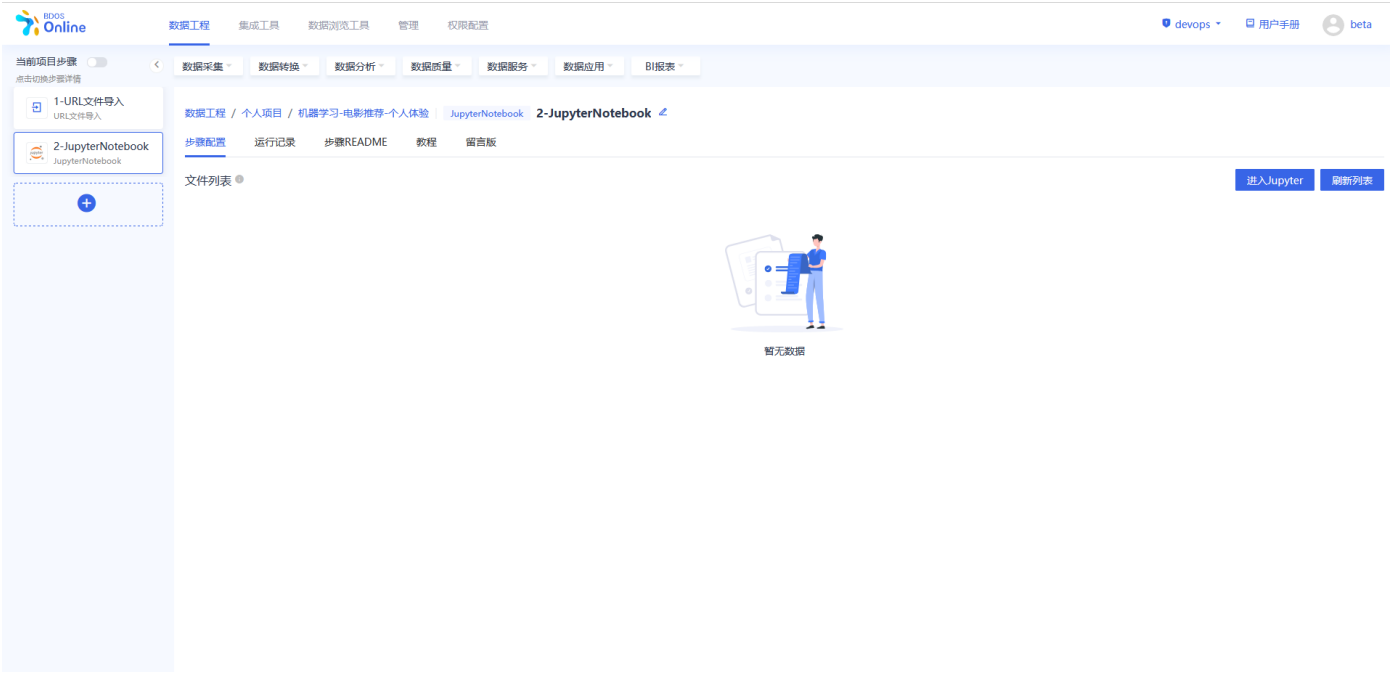
从当前项目步骤中进行添加，点击各个目录中的具体操作，依照该方法，分别添加以下几个步骤：

1.数据采集--URL文件导入

2.数据分析--jupyterNotebook



添加完成后，对步骤进行修改名称，以区分，可以参照下图进行修改。



步骤三：URL 文件导入

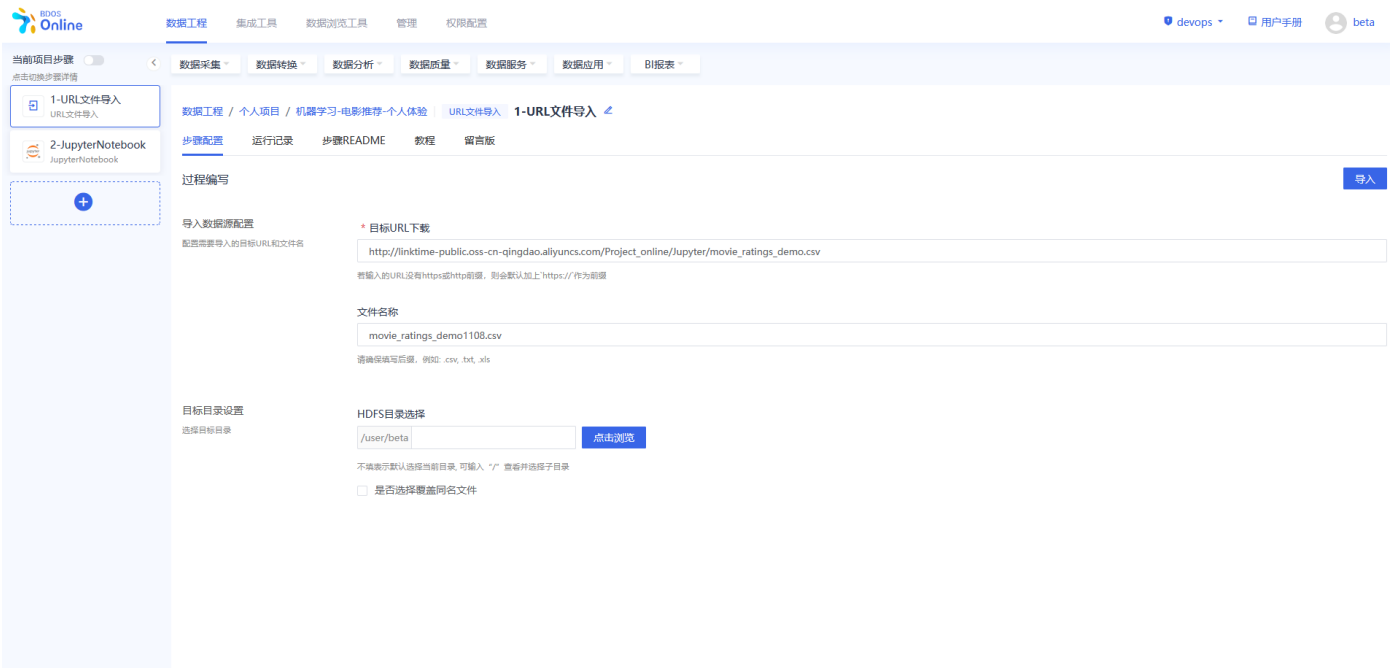
在BDOS Online大数据平台，用户可通过URL文件导入，导入到系统的HDFS。

电影名称数据集Web下载路径：http://linktime-public.oss-cn-qingdao.aliyuncs.com/Project_online/Jupyter/movies_metadata.csv

电影评分数据集Web下载路径：http://linktime-public.oss-cn-qingdao.aliyuncs.com/Project_online/Jupyter/movie_ratings_demo.csv

文件名称：xxx.csv（用户选填，不填则默认为url链接包含的文件名，用户也可参照示例进行名称自定义并带上文件后缀）

HDFS目录选择：保持默认



步骤四：对数据进行处理和导出至Hive

进入JupyterLab，新建PySpark notebook，并在PySpark程序步骤对电影测评demo数据进行处理并导出至Hive。

步骤一操作

> 注：
>
> `机构项(xxx/xxx为org/xxx)` 1.替换org/xxx的xxx为机构名称 2.若URL文件导入步骤填入自定义名称，则替换csv为实际csv表名，否则不用修改csv表名
>
> `个人项(xxx/xxx为user/xxx)` 1.替换user/xxx的xxx为当前登录用户名 2.若URL文件导入步骤填入自定义名称，则替换csv为实际csv表名，否则不用修改csv**表名**

```
data=spark.read.csv('hdfs://default/user/beta/movie_ratings_demo1108.csv',header=True)
name=spark.read.csv('hdfs://default/user/beta/movies_metadata1108.csv', header=True)
name=name.select("id","title")
for col in data.columns:
    data = data.withColumnRenamed(col, col.lower())
for col in name.columns:
    name = name.withColumnRenamed(col, col.lower())
data.show(1)
name.show(1)
步骤一说明
```

> 1.以PySpark的格式读取导入到Hive目标表的实验数据到PySpark dataframe
>
> 2.使用function data.show() 将dataframe 的内容进行展示。如：show（1），即展示数据集第1行，不填则默认展示前20行。

```
步骤二操作
data = data.join(name, data.movieid == name.id)
data=data.select("userid","movieid","rating","timestamp","title")
data.show(1)
步骤二说明
```

> 将电影评分数据集与电影名称数据集进行关联

```
步骤三操作
data=data.distinct()
data = data.dropDuplicates(subset=[c for c in data.columns if c in
["userid","movieid"]])
data.count()
步骤三说明
```

> 1.筛选重复数据，如：使用function data.distinct() 对完全相同的行进行去重。

> 2.进一步筛选重复数据，如：在 userId和movieId相同的情况下，只允许一条评分存在。

> 3.数据统计，如：使用function data.count() 统计数据集行数。

步骤四操作

```
import pyspark.sql.functions as f
data.agg(*[(1-(f.count(c) /f.count('*'))).alias(c+'_missing') for c in
data.columns]).show()
data=data.na.drop(subset=['rating'])
data=data.dropna(thresh=2)
```

步骤四说明

> 1.打印每列数据的空缺比。

> 2.删除Class项结果项空缺的数据，如:(data.na.drop(subset=['rating']))，对“rating”项缺失的行进行删除。

> 3.删除非Class项空缺数>=2的行数据，如对 thresh 进行参数设置来控制判断空缺项的阈值。

步骤五操作

```
data=data.withColumnRenamed('userid', 'user')
data=data.withColumnRenamed('movieid', 'movie')
data
```

步骤五说明

> 对列进行重命名,如：使用function withColumnRenamed（）对列进行自定义的命名。

步骤六操作

```
data= data.withColumn("user",data['user'].cast('int'))
data= data.withColumn("rating",data['rating'].cast('int'))
data= data.withColumn("movie",data["movie"].cast('int'))
```

步骤六说明

> 1.对数据类型进行转换，如：使用 function withColumn() 对数据类型进行转换，将string类型数据转换成double类型数据。

步骤七操作

```
from pyspark.sql.functions import col
data.groupBy("rating").count().orderBy(col("count").desc()).show(truncate=False)
data=data.filter(( data.rating== 0)|( data.rating== 1)|(data.rating == 2) |
(data.rating == 3) | (data.rating == 4)|(data.rating == 5))
```

步骤七说明

> 筛选满足打分要求的结果项，即通过 function filter（）筛选 rating 项为0或1的数据。

步骤八操作

> 注：

> 机构项目(xxx.xxx为org_xxx)

1.替换org_xxx的xxx为机构名称 2.table1为Hive目标表名，用户可自定义目标表名。

> 个人项目(xxx_xxx为user_xxx)

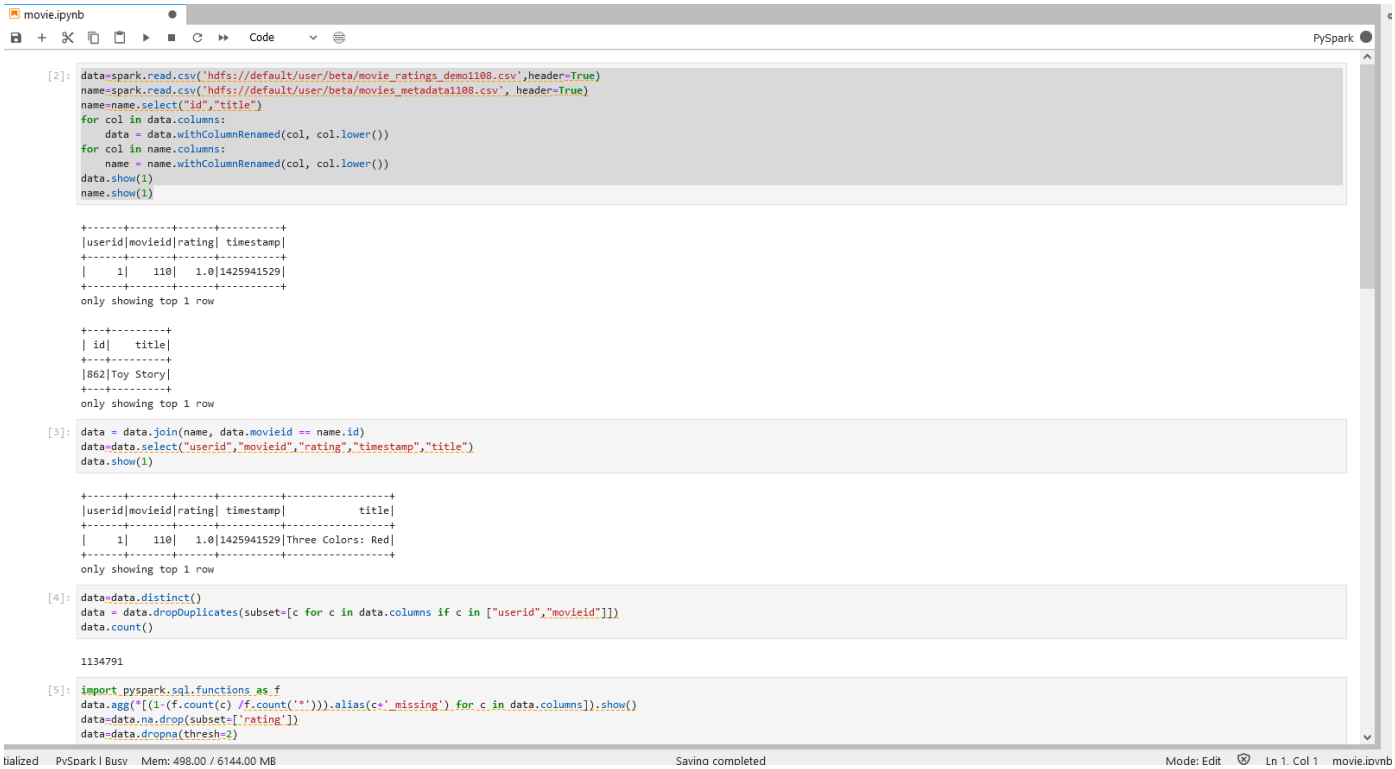
> 1.替换user_xxx的xxx为当前登录用户名 2.table1为Hive目标表名，用户可自定义目标表名。

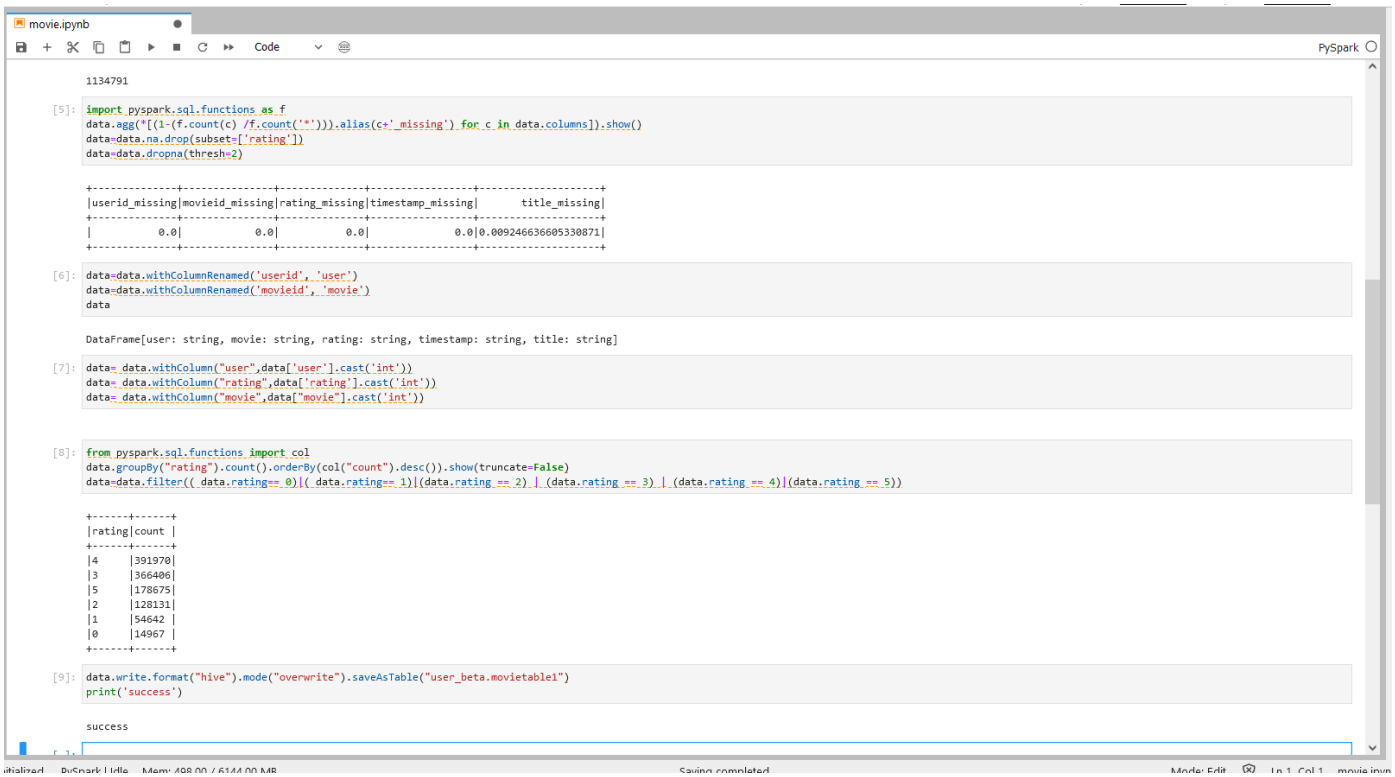
```
data.write.format("hive").mode("overwrite").saveAsTable("xxx_xxx.table1")
print('success')
```

步骤八说明

> 将结果数据导出到目标Hive库表

具体操作可以参照下图。





步骤五：ALS模型预测与调参

再次在JupyterLab中新建PySpark notebook，在PySpark程序中进行ALS模型预测与调参。

步骤一操作

> 注：`机构项目(xxx_xxx为org_xxx)`
>
> 1.替换org_xxx的xxx为机构名称 2.table2为实际Hive目标表名
>
> `个人项(xxx_xxx为user_xxx)` 1.替换user_xxx的xxx为当前登录用户名 2.table2为实际Hive目标表名

```
df=spark.sql("select * from xxx_xxx.table1")
(train, test) = df.randomSplit([0.8, 0.2],seed = 11)
print('success')
```

步骤一说明

> 导入上一个步骤的输出到Jupyter，并将数据分成测试集和训练集。

步骤二操作

```
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator
als=
ALS(userCol="user",itemCol="movie",ratingCol="rating",nonnegative=True,implicitPrefs=Fa
lse, coldStartStrategy="drop",maxIter=10)
als_evaluator = RegressionEvaluator(predictionCol="prediction", \
```

```
                labelCol="rating",metricName="rmse")
param_grid = ParamGridBuilder() \
            .addGrid(als.regParam, [ .01]) \
            .build()
#, .05, .1, .15
tvs = TrainValidationSplit(estimator=als,
                           estimatorParamMaps=param_grid,
                           evaluator=als_evaluator,
                           trainRatio = 0.8)
model=tvs.fit(train)
model=model.bestModel
als_prediction=model.transform(test)
evaluator = RegressionEvaluator(metricName="mae", labelCol="rating",
                                predictionCol="prediction")

mae = evaluator.evaluate(als_prediction)
print("mean absolute error = " + str(mae))
步骤二说明
```

> ALS模型的调参与预测
>
> - 运用TrainValidationSplit（TVS）来进行参数调优。
> - 运用param_grid方程来定义TVS检测的参数。

```
步骤三操作
```

> 注：

> 机构项目： 1."hdfs:///xxx/xxx/data/mllib/alsmodel" 为`hdfs:///org/xxx/data/mllib/alsmodel`，xxx替换为当前机构名。 2."xxx_xxx.table2"为org_xxx，xxx替换为当前机构名，table2为Hive目标表名，用户可自定义目标表名。

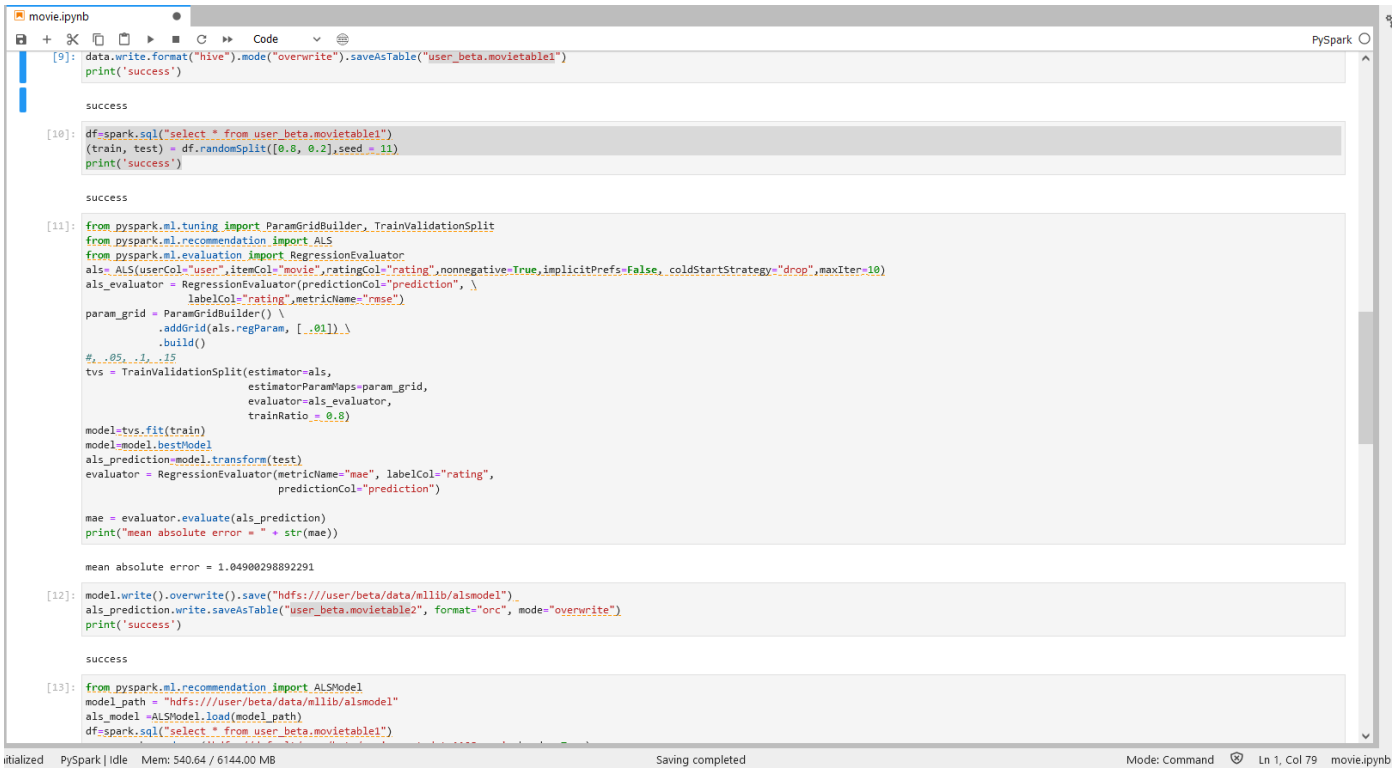
> 个人项目： 1."hdfs:///xxx/xxx/data/mllib/alsmodel" 为`hdfs:///user/xxx/data/mllib/alsmodel`，xxx替换为当前登录用户名。 2."xxx_xxx.table2"为user_xxx，xxx替换为当前登录用户名，table2为Hive目标表名，用户可自定义目标表名。

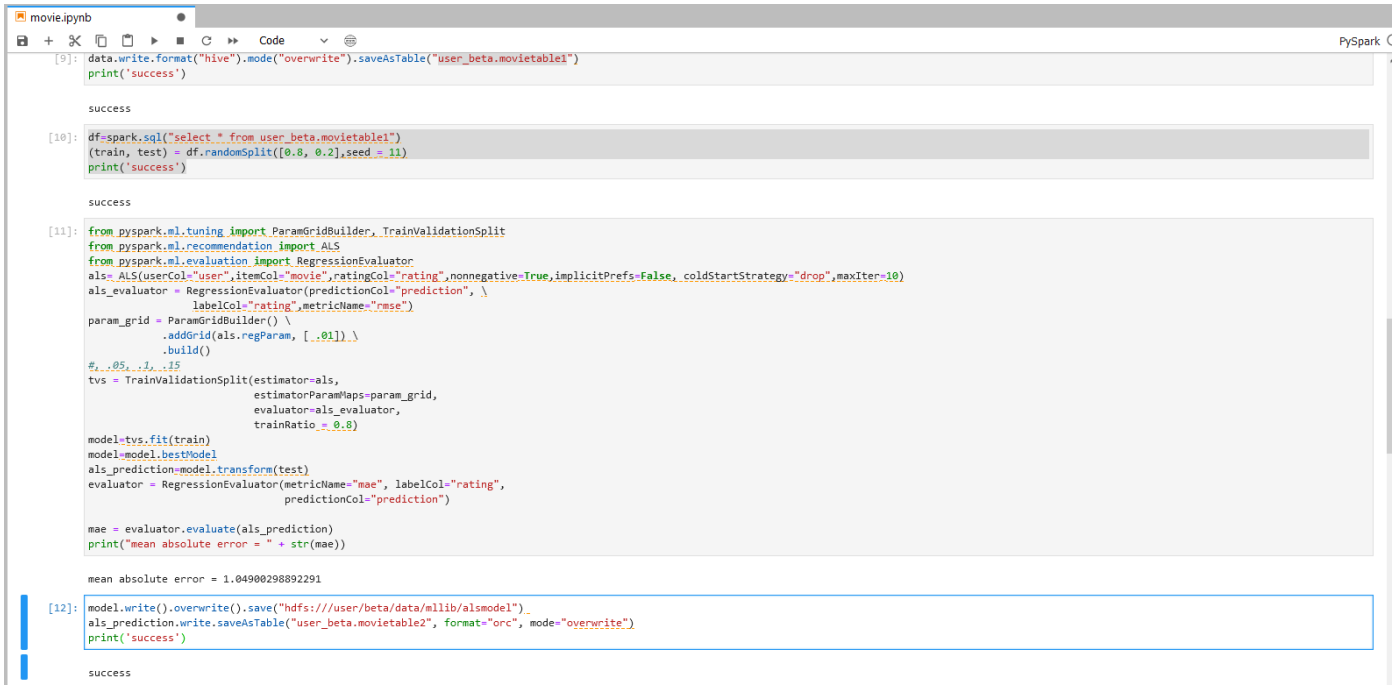
```
model.write().overwrite().save("hdfs:///xxx/xxx/data/mllib/alsmodel")
als_prediction.write.saveAsTable("xxx_xxx.table2", format="orc", mode="overwrite")
print('success')

步骤三说明
```

> 将训练完成的最佳模型存入HDFS以便后续使用。

> 将最佳预测结果存入Hive目标表中。

具体操作可以参照下图。





步骤六：电影个性化推荐

再次进入JupyterLab中新建PySpark notebook，在PySpark程序中进行电影个性化推荐。

步骤一操作

> 注：

> 机构项目： 1."hdfs:///xxx/xxx/data/mllib/alsmodel" 为`hdfs:///org/xxx/data/mllib/alsmodel`，xxx替换为当前机构名。 2."xxx_xxx.table1"为org_xxx，xxx替换为当前机构名，table1为Hive目标表名，用户可自定义目标表名。 3."xxx/xxx/movies_metadata.csv'"为org/xxx，xxx替换为当前机构名，movies_metadata.csv替换为用户自定义的hfds表名。

> 个人项目: 1."hdfs:///xxx/xxx/data/mllib/alsmodel" 为`hdfs:///user/xxx/data/mllib/alsmodel`，xxx替换为当前登录用户名。 2."xxx_xxx.table1"为user_xxx，xxx替换为当前登录用户名，table1为Hive目标表名，用户可自定义目标表名。 3."xxx/xxx/movies_metadata.csv'"为user/xxx，xxx替换为当前登录用户名，movies_metadata.csv替换为用户自定义的hfds表名。

```
from pyspark.ml.recommendation import ALSModel
model_path = "hdfs:///xxx/xxx/data/mllib/alsmodel"
als_model =ALSModel.load(model_path)
df=spark.sql("select * from xxx_xxx.table1")
name=spark.read.csv('hdfs://default/xxx/xxx/movies_metadata.csv', header=True)
print('success')
步骤一说明
```

> 导入上一个步骤的输出数据到Jupyter。

```
步骤二操作
from pyspark.sql.functions import rand
df.orderBy(rand()).limit(1).select("user").show()
步骤二说明
```

> 从数据集中随机选择一个用户id，以便于后续进行检测。在limit（）的括号中填入选择的数量。

```
步骤三操作
recommend=als_model.recommendForAllUsers(10)
recommend.select("recommendations").show(1)
def recommendforuser(userid):
    res=recommend.filter(recommend.user==userid)
    res=res.select("recommendations")
    res=res.take(1)[0]["recommendations"]
    tname=[]
    for i in res:
        temp=name.filter(name.id==i.movie)
        n=temp.select("title").take(1)[0]["title"]
        tname.append(n)
    return tname
print('success')
步骤三说明 - 运用模型对用户进行推荐
```

> 使用function recommendForAllUsers（）方程，在括号中填入每个用户需要推荐的电影数量。 使用recommendforuser（）这个方程，将用户id对应推荐的电影id转化为电影名称，使推荐更直观。

步骤四操作

```
userid=21260
recommendforuser(userid)
```

步骤四说明

> 推荐实例，在function recommendforuser()括号中填入用户id，则会得到一个包含十部电影名称的队列。如示例中随机填入userid“21260”，会返回对应的推荐队列。

步骤五操作

> 注：

机构项目：

> "xxx_xxx.table3"为org_xxx，xxx替换为当前机构名，table3为Hive目标表名，用户可自定义目标表名。

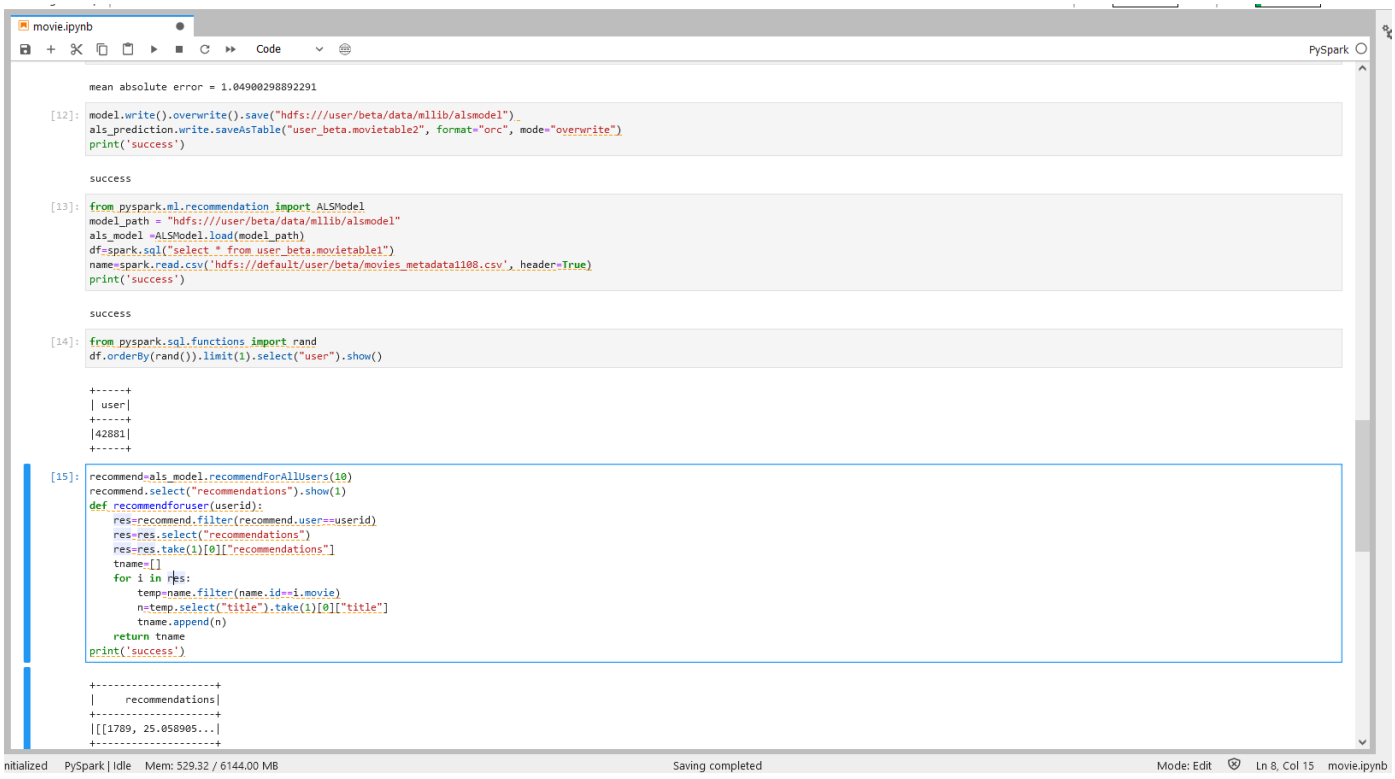
> 个人项目： "xxx_xxx.table3"为user_xxx，xxx替换为当前登录用户名，table3为Hive目标表名，用户可自定义目标表名。

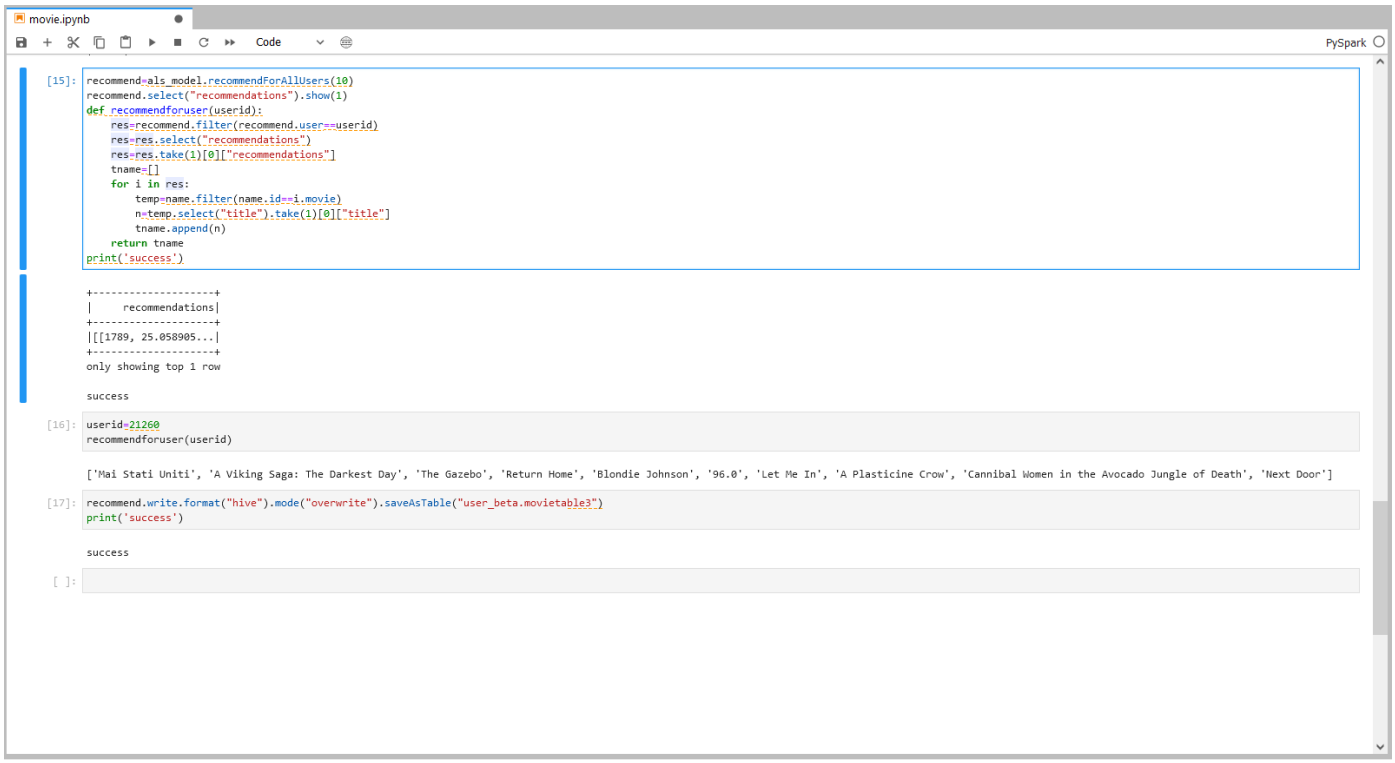
```
recommend.write.format("hive").mode("overwrite").saveAsTable("xxx_xxx.table3")
print('success')
```

步骤五说明

> 将最佳预测结果存入目标 Hive 表中。

具体操作可参照下图。





备注：建议完成整个步骤后将该项工程暂停，可参考下图。