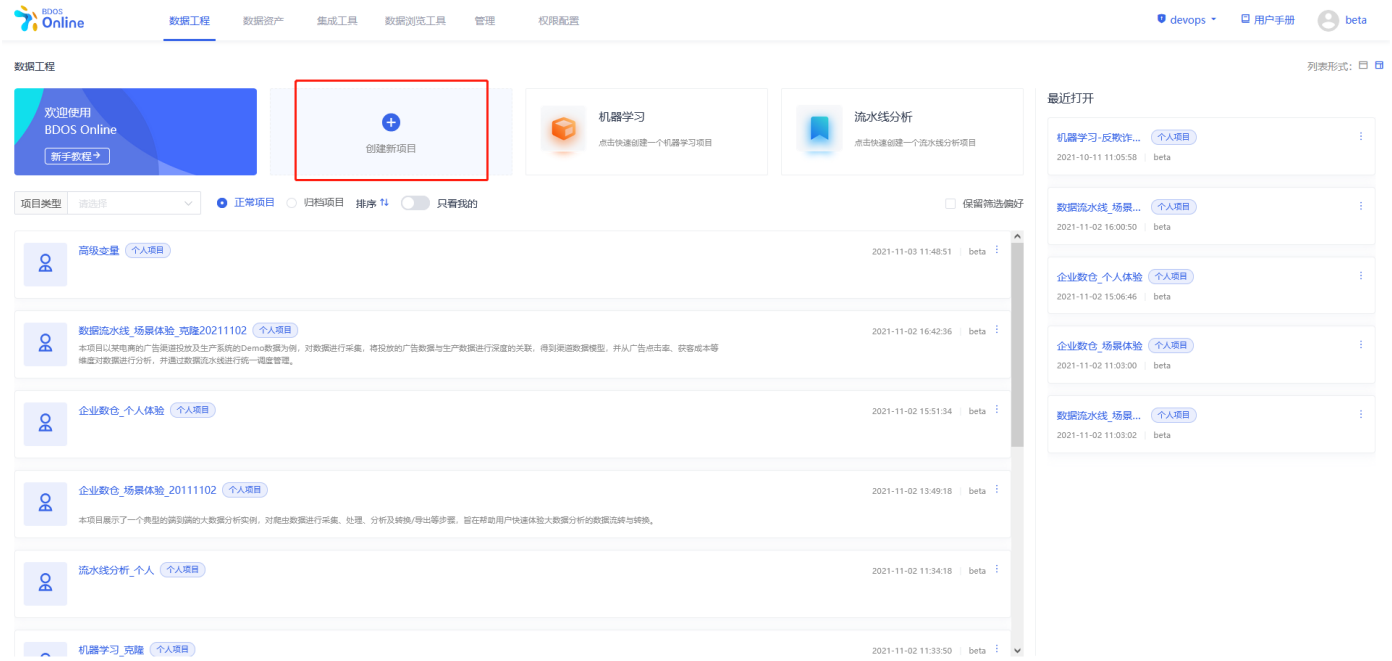
机器学习-美食评分预测

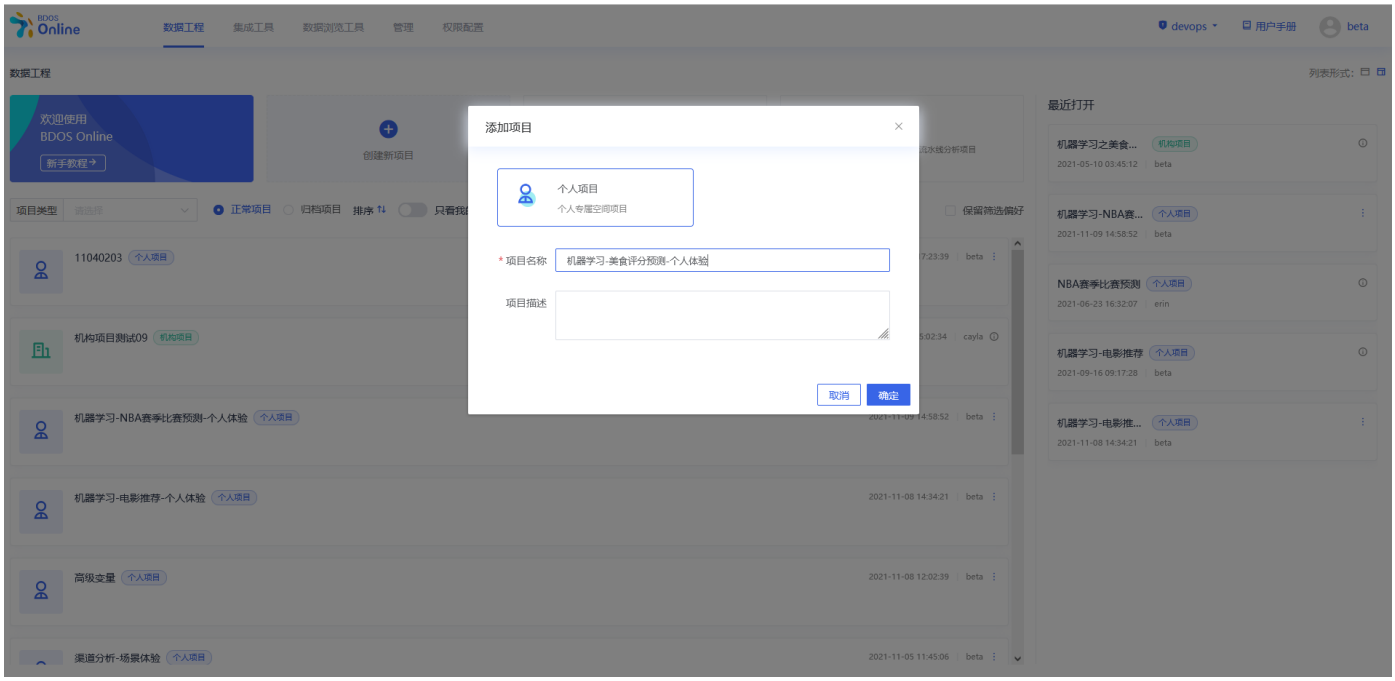
本项目展示了对一个既包含文字类信息又包含数字型信息的数据集，通过线性回归模型对用户评分进行预测的机器学习过程。原始数据来源：foodreview。数据集包含了1999年10月到2012年10月，Amozon用户对于美食的评论数据。

本介绍以视频的形式，向大家展示一个机器学习通过线性回归模型进行用户评分预测的实例，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

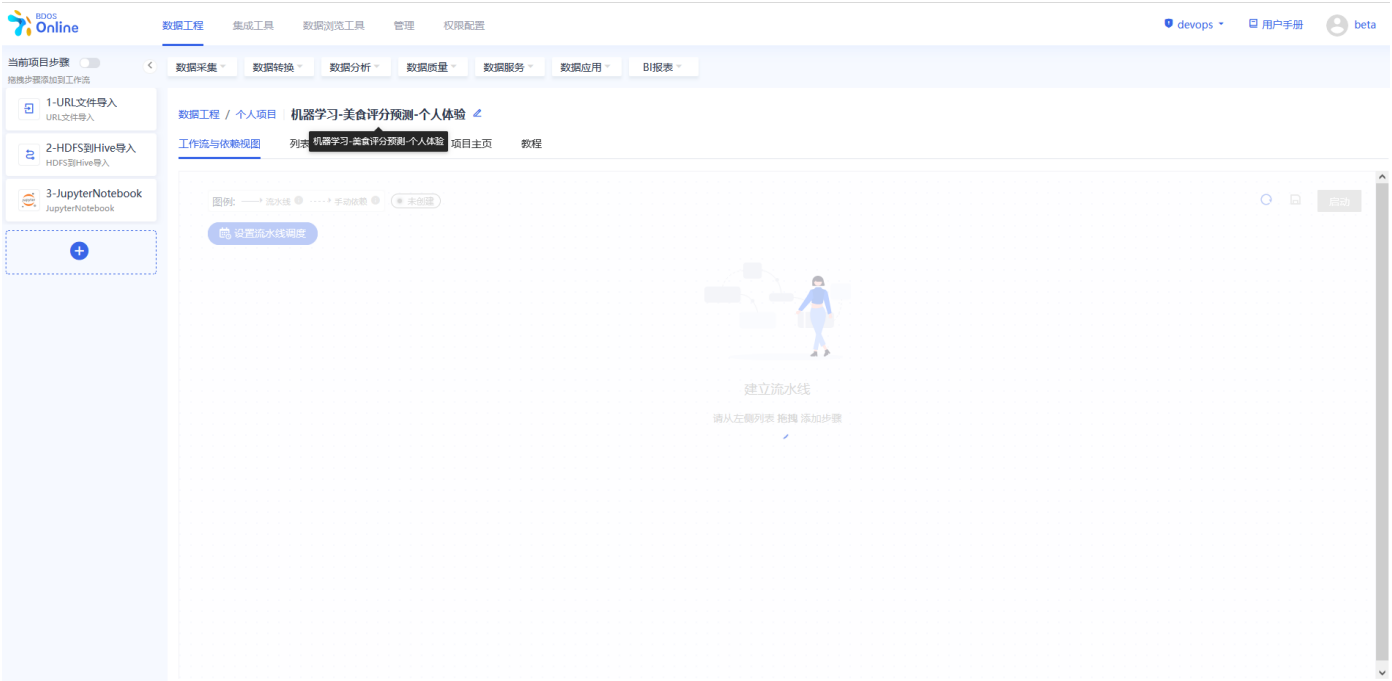
从当前项目步骤中进行添加，点击各个目录中的具体操作，依照该方法，分别添加以下几个步骤：

1.数据采集--URL文件导入

2.数据转换--HDFS到Hive导入

3.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

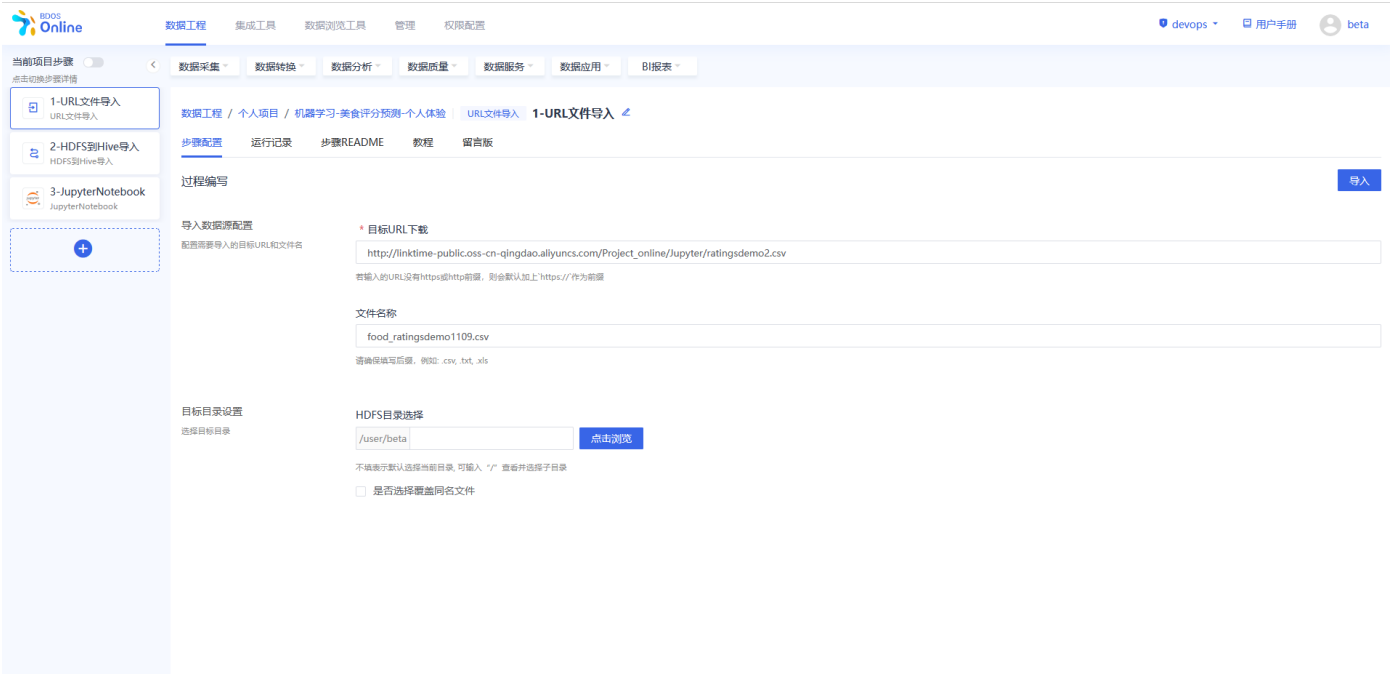
在BDOS Online大数据平台，用户可通过URL文件导入，导入分析数据到系统的HDFS。

`Web下载路径：` http://linktime-public.oss-cn-qingdao.aliyuncs.com/Project_online/Jupyter/ratingsdemo2.csv

`文件名称：` xxx.csv（用户参照示例进行名称自定义并带上文件后缀）

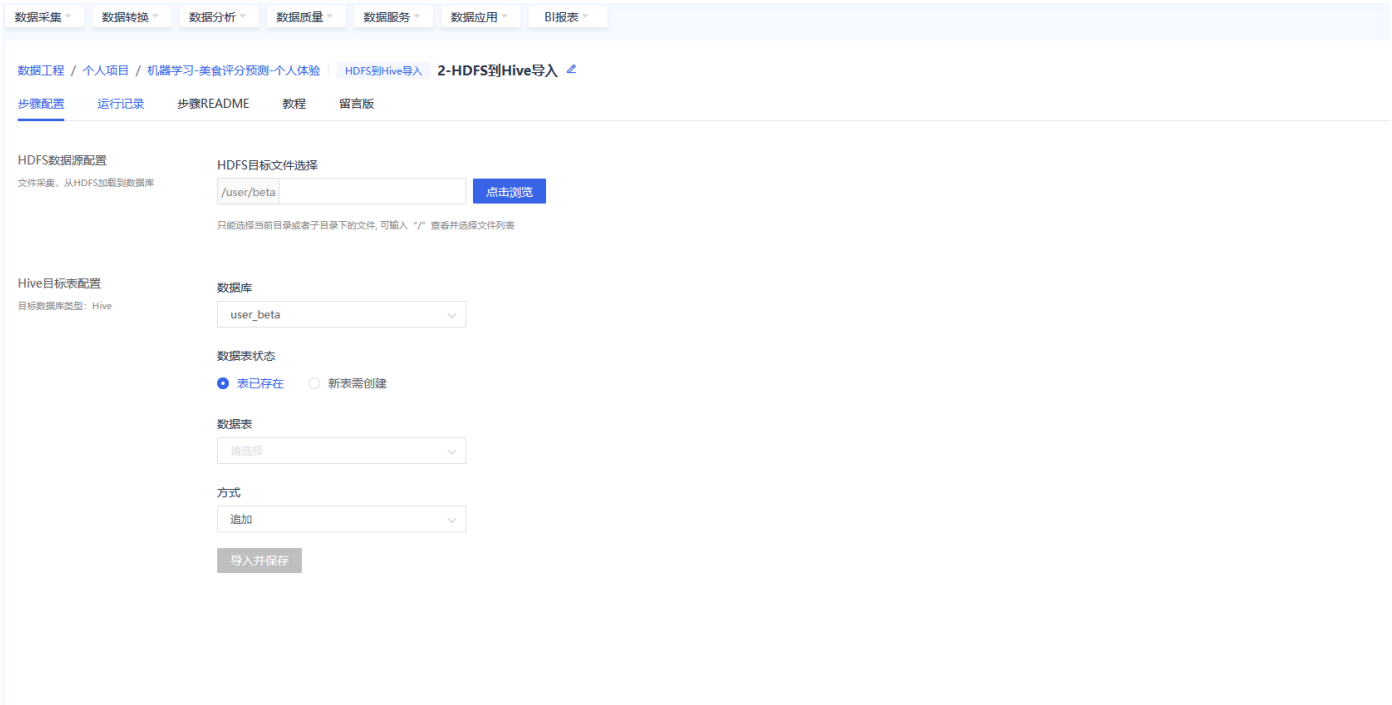
`HDFS目录选择：` 保持默认

具体操作可以参考下图。

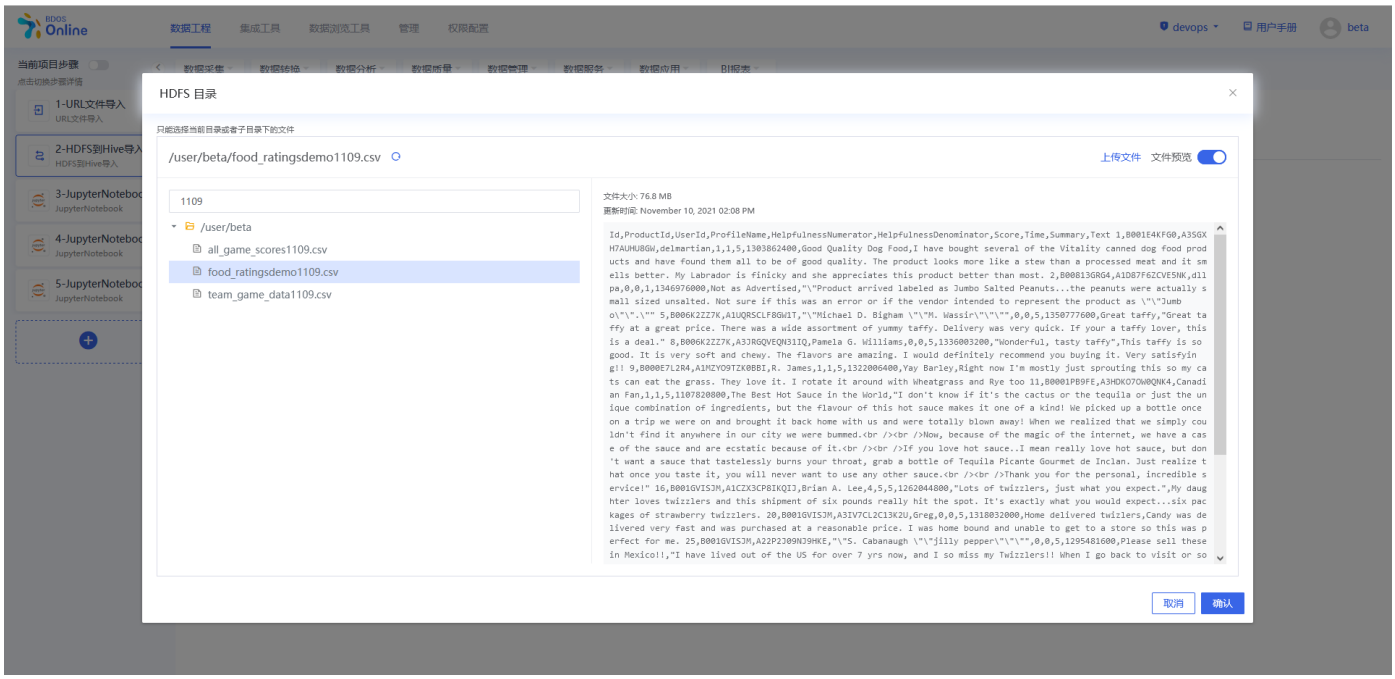


步骤四：将数据导入到目标 Hive 库

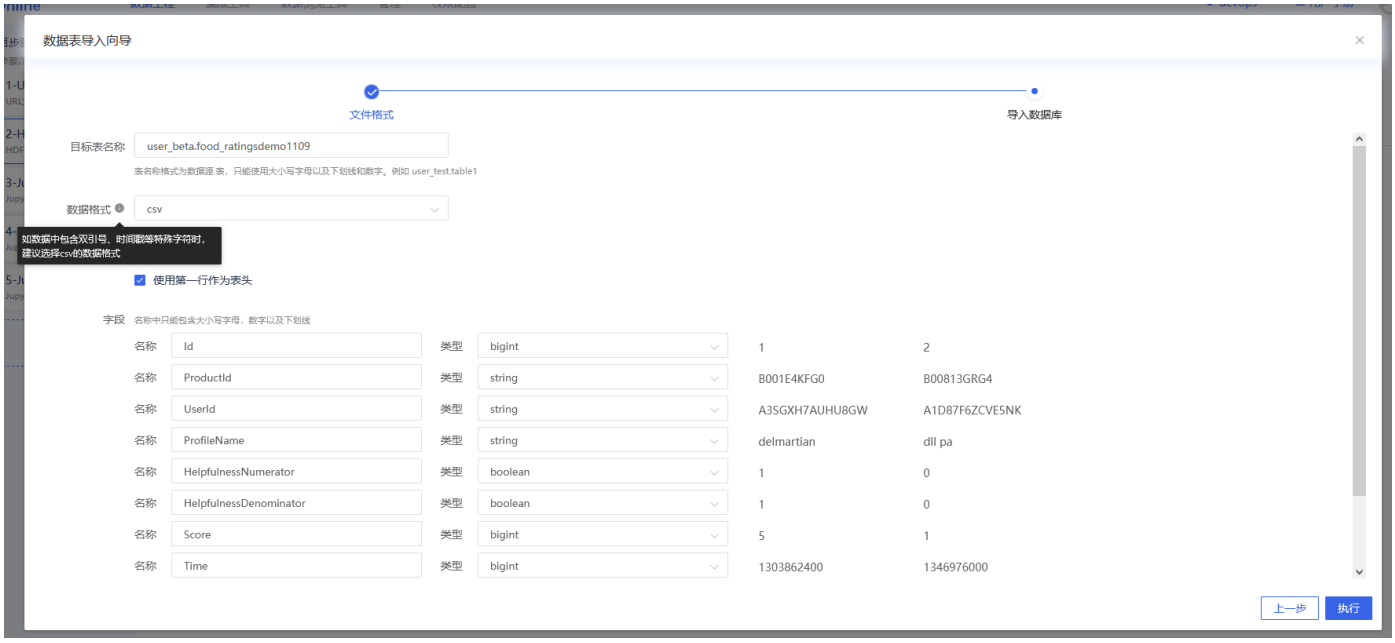
在BDOS Online大数据平台，用户可通过HDFS 到 Hive 导入步骤把数据导入到目标 Hive 库。



选择点击浏览，进入HDFS目录中，选择上一步保存名字的文件，点击确认。



Hive目标表配置：这里以新表需创建为例，点击`打开向导`。数据格式选择csv，点击执行。



执行结果可以在运行记录中查看。

步骤五：对数据进行处理和导出

进入JupyterLab，新建PySpark notebook，并在PySpark程序步骤对美食测评demo数据进行处理并导出。

> 进入jupyterlab本项目选的PySpark环境

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.替换table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.替换table为实际Hive目标表名

```
data=spark.sql("select * from xxx_xxx.table")
data=data.filter(data.text!="Text")
data.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：通过在show（）中填写数字，选择需要展示的行数，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.distinct()
data = data.dropDuplicates(subset=[c for c in data.columns if c in ["productid",
"userid","time"]])
data.count()
```

步骤二说明

> 1.处理重复数据，如：使用function data.distinct() 进行去重。

> 2.处理 productid，userid和 time 相同情况下的重复数据。

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

步骤三操作

```
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=['score'])
data=data.dropna(thresh=3)
```

步骤三说明

> 1.打印每列数据的空缺比，并删除空缺比高的列。（本示例数据集无空缺比高的列）。

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

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

步骤四操作

```
data = data.withColumn("helpfulnessnumerator",data['helpfulnessnumerator'].cast('int'))
data =
data.withColumn("helpfulnessdenominator",data['helpfulnessdenominator'].cast('int'))
data = data.withColumn("score",data['score'].cast('int'))
data = data.withColumn("time",data['time'].cast('int'))
print('success')
```

步骤四说明

> 1.使用 function withColumn() 进行数据类型转换，将string类型数据转换成int类型数据。

步骤五操作：

```
from pyspark.ml.feature import StringIndexer
product_indexer = StringIndexer(inputCol='productid', outputCol='product').fit(data)
data= product_indexer.transform(data)
user_indexer = StringIndexer(inputCol='userid', outputCol='user').fit(data)
data= user_indexer.transform(data)
print('success')
```

步骤五说明：

> 使用 StringIndexer 将类别型数值转化为数字型数值

步骤六操作

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

步骤六说明：

> 筛选满足打分要求的结果项，即通过function filter（）筛选 score 项为0-5的数据。

```
步骤七操作
from pyspark.sql.functions import regexp_replace
data = data.withColumn("only_str",regexp_replace(col('summary'), '\d+', ''))
print('success')
步骤七说明
```

> 使用function regexp_replace 删除非文字数据，如：删除数字、表情等数据。

```
步骤八操作：
from pyspark.ml.feature import  RegexTokenizer,StopWordsRemover
regex_tokenizer = RegexTokenizer(inputCol="only_str", outputCol="nwords",
pattern="\\W")
data = regex_tokenizer.transform(data)
remover= StopWordsRemover(inputCol="nwords", outputCol="filtered")
data = remover.transform(data)
data.select("nwords","filtered").show()
步骤八说明：
```

> 删除非行为数据（如：删除介词、代词等小词）

```
步骤九操作
```

> 注：
>
> > 机构项(xxx_xxx为org_xxx)
>
> 1.替换org_xxx的xxx为机构名称 2.替换table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.替换table为实际Hive目标表名

```
data.write.format("hive").mode("overwrite").saveAsTable("xxx_xxx.table2")
print('success')
步骤九说明
```

> 将结果数据导出到目标Hive库表

具体操作及结果可以参照下图。

foodrating1.ipynb

Code

PySpark

[6]:
```
data=spark.sql("select * from user_beta.food_ratingsdemo1110")
data=data.filter(data.text!="Text")
data.show(1)
```

```
+---+----------+--------------+-----------+--------------------+---------------------+-----+----------+--------------------+--------------------+
| id| productid|        userid|profilename|helpfulnessnumerator|helpfulnessdenominator|score|      time|             summary|                text|
+---+----------+--------------+-----------+--------------------+---------------------+-----+----------+--------------------+--------------------+
|  1|B001E4KFG0|A3SGXH7AUHU8GW| delmartian|                   1|                    1|    5|1303862400|Good Quality Dog ...|I have bought sev...|
+---+----------+--------------+-----------+--------------------+---------------------+-----+----------+--------------------+--------------------+
only showing top 1 row
```

[7]:
```
data=data.distinct()
data = data.dropDuplicates(subset=[c for c in data.columns if c in ["productid", "userid","time"]])
data.count()
```

```
170405
```

[8]:
```
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=['score'])
data=data.dropna(thresh=3)
```

```
+----------+----------------+-------------+----------------+-----------------------+-------------------------+-------------------------------+-------------+-----------+---------------+------------+
|id_missing|productid_missing|userid_missing|profilename_missing|helpfulnessnumerator_missing|helpfulnessdenominator_missing|score_missing|time_missing|summary_missing|text_missing|
+----------+----------------+-------------+----------------+-----------------------+-------------------------+-------------------------------+-------------+-----------+---------------+------------+
|       0.0|             0.0|          0.0|             0.0|                    0.0|                      0.0|                            0.0|          0.0|        0.0|            0.0|         0.0|
+----------+----------------+-------------+----------------+-----------------------+-------------------------+-------------------------------+-------------+-----------+---------------+------------+
```

[9]:
```
data = data.withColumn("helpfulnessnumerator",data['helpfulnessnumerator'].cast('int'))
data = data.withColumn("helpfulnessdenominator",data['helpfulnessdenominator'].cast('int'))
data = data.withColumn("score",data['score'].cast('int'))
data = data.withColumn("time",data['time'].cast('int'))
print('success')
```

```
success
```

[10]:
```
from pyspark.sql.functions import col
data.groupBy("score").count().orderBy(col("count").desc()).show(truncate=False)
data=data.filter((data.score == 0)|(data.score == 1)|(data.score == 2) | (data.score == 3) | (data.score == 4)|(data.score == 5))
```

```
+-----+------+
|score|count |
+-----+------+
|5    |108269|
+-----+------+
```

foodrating1.ipynb

Code

PySpark

[10]:
```
from pyspark.sql.functions import col
data.groupBy("score").count().orderBy(col("count").desc()).show(truncate=False)
data=data.filter((data.score == 0)|(data.score == 1)|(data.score == 2) | (data.score == 3) | (data.score == 4)|(data.score == 5))
```

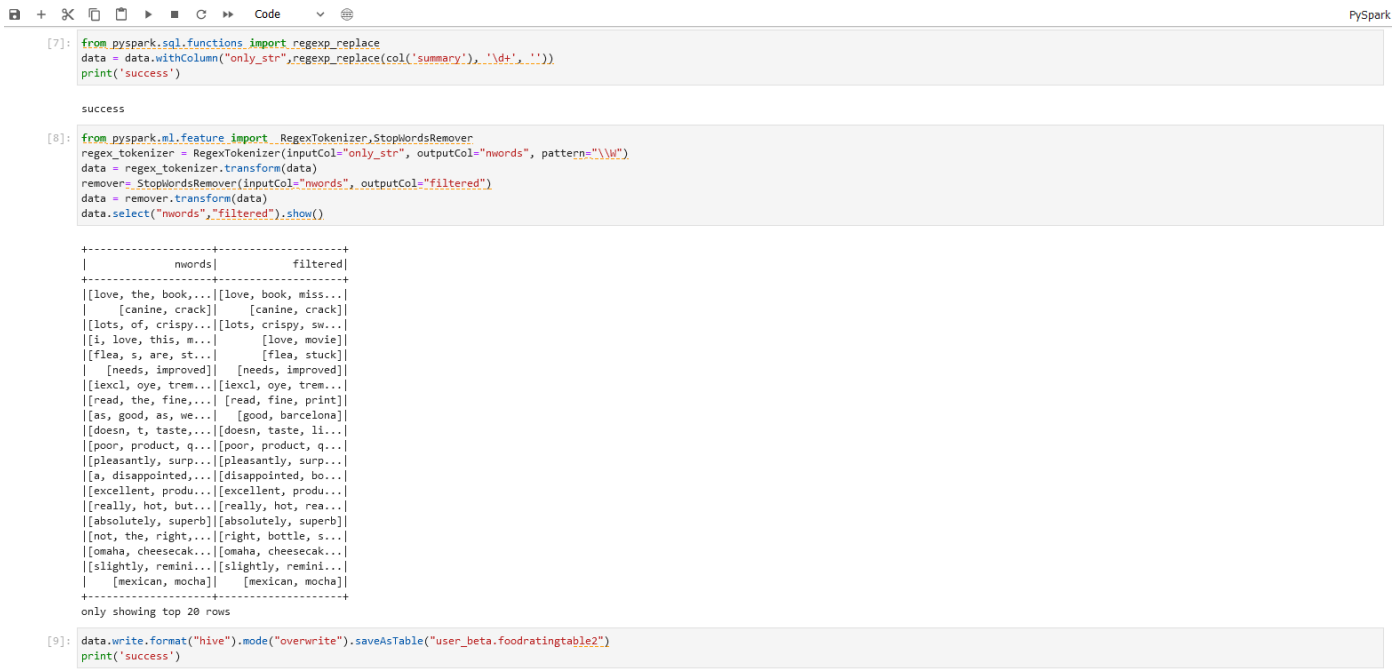
```
+-----+------+
|score|count |
+-----+------+
|5    |108269|
|4    |24044 |
|1    |15822 |
|3    |12748 |
|2    |9037  |
|0    |327   |
|null |78    |
|6    |14    |
|10   |6     |
|8    |5     |
|7    |5     |
|9    |5     |
|47   |4     |
|14   |4     |
|21   |3     |
|16   |3     |
|19   |3     |
|15   |3     |
|24   |2     |
|33   |2     |
+-----+------+
only showing top 20 rows
```

[11]:
```
from pyspark.sql.functions import regexp_replace
data = data.withColumn("only_str",regexp_replace(col('summary'), '\d+', ''))
print('success')
```

```
success
```

[12]:
```
from pyspark.ml.feature import  RegexTokenizer,StopWordsRemover
regex_tokenizer = RegexTokenizer(inputCol="only_str", outputCol="nwords", pattern="\\W")
data = regex_tokenizer.transform(data)
remover= StopWordsRemover(inputCol="nwords", outputCol="filtered")
data = remover.transform(data)
data.select("nwords","filtered").show()
```

```
+--------------------+--------------------+
|              nwords|            filtered|
+--------------------+--------------------+
```



```
[7]: from pyspark.sql.functions import regexp_replace
data = data.withColumn("only_str",regexp_replace(col('summary'), '\d+', ''))
print('success')

success

[8]: from pyspark.ml.feature import  RegexTokenizer,StopWordsRemover
regex_tokenizer = RegexTokenizer(inputCol="only_str", outputCol="nwords", pattern="\\W")
data = regex_tokenizer.transform(data)
remover= StopWordsRemover(inputCol="nwords", outputCol="filtered")
data = remover.transform(data)
data.select("nwords","filtered").show()

+--------------------+--------------------+
|              nwords|            filtered|
+--------------------+--------------------+
|[love, the, book,...|[love, book, miss...|
|     [canine, crack]|     [canine, crack]|
|[lots, of, crispy...|[lots, crispy, sw...|
|[i, love, this, m...|       [love, movie]|
|[flea, s, are, st...|       [flea, stuck]|
|   [needs, improved]|   [needs, improved]|
|[iexcl, oye, trem...|[iexcl, oye, trem...|
|[read, the, fine,...| [read, fine, print]|
|[as, good, as, we...|   [good, barcelona]|
|[doesn, t, taste,...|[doesn, taste, li...|
|[poor, product, q...|[poor, product, q...|
|[pleasantly, surp...|[pleasantly, surp...|
|[a, disappointed,...|[disappointed, bo...|
|[excellent, produ...|[excellent, produ...|
|[really, hot, but...|[really, hot, rea...|
|[absolutely, superb]|[absolutely, superb]|
|[not, the, right,...|[right, bottle, s...|
|[omaha, cheesecak...|[omaha, cheesecak...|
|[slightly, remini...|[slightly, remini...|
|    [mexican, mocha]|    [mexican, mocha]|
+--------------------+--------------------+
only showing top 20 rows

[9]: data.write.format("hive").mode("overwrite").saveAsTable("user_beta.foodratingtable2")
print('success')
```

步骤六：特征提取

延续上一步的PySpark notebook，在PySpark程序中运用文本特征提取模型，将用户评论数据转换为特征向量。

> 进入jupyterlab本项目选的PySpark环境

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.替换table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.替换table2为实际Hive目标表名

```
df=spark.sql("select * from xxx_xxx.table2")
print('success')
```

步骤一说明 - 数据转换与导入

> 导入上一个步骤的输出到Jupyter。

步骤二操作

```
vocabsize=5000
from pyspark.ml.feature import NGram, VectorAssembler
def build_ngrams_wocs(inputCol=["only_str","score"], n=3):
    ngrams = [
        NGram(n=i, inputCol="filtered", outputCol="{0}_grams".format(i))
        for i in range(1, n + 1)
    ]
```

```
    cv = [
        CountVectorizer(vocabSize=vocabsize,inputCol="{0}_grams".format(i),
            outputCol="{0}_tf".format(i))
        for i in range(1, n + 1)
    ]
    idf = [IDF(inputCol="{0}_tf".format(i), outputCol="{0}_tfidf".format(i),
minDocFreq=5) for i in range(1, n + 1)]

    assembler = [VectorAssembler(
        inputCols=["{0}_tfidf".format(i) for i in range(1, n + 1)],
        outputCol="featuresn",
    )]
    return Pipeline(stages=ngrams + cv + idf+ assembler)
print('success')
from pyspark.ml.feature import Tokenizer,CountVectorizer,IDF
from pyspark.ml import Pipeline
trigramwocs_pipelineFit = build_ngrams_wocs().fit(df)
df= trigramwocs_pipelineFit.transform(df)
df = df.na.fill(0)
print('success')
步骤二说明 - 文本特征提取
```

> 从unigram，bigram，trigram中分别获得5000个特征，经过合并后，featuresn会得到15000个特征。

```
步骤三操作
from pyspark.ml.feature import StringIndexer
va = VectorAssembler(inputCols=
["user","product","helpfulnessnumerator","helpfulnessdenominator","time","featuresn"],
outputCol="newnfeatures")
df= va.transform(df)
print('success')
步骤三说明 - 合并文本类特征与数据类特征
```

> 运用function VectorAssembler将文本特征vector和数据特征相结合。

```
步骤四操作
```

> 注： 机构项目时：

> "xxx_xxx.table3"为org_xxx，xxx替换为当前机构名，table3替换为用户自定义的Hive表名。

> 个人项目时

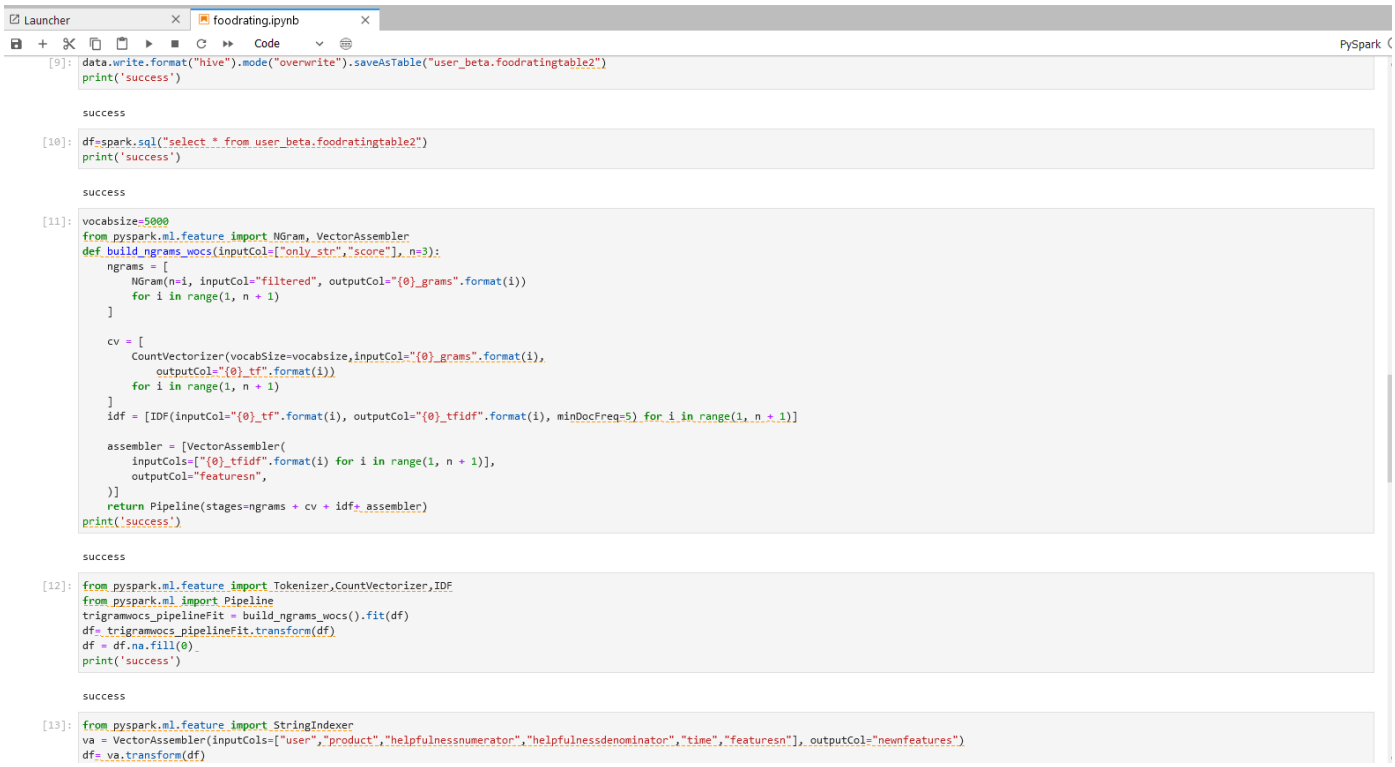
> "xxx_xxx.table3"为user_xxx，xxx替换为当前登录用户名，table3替换为用户自定义的Hive表名。

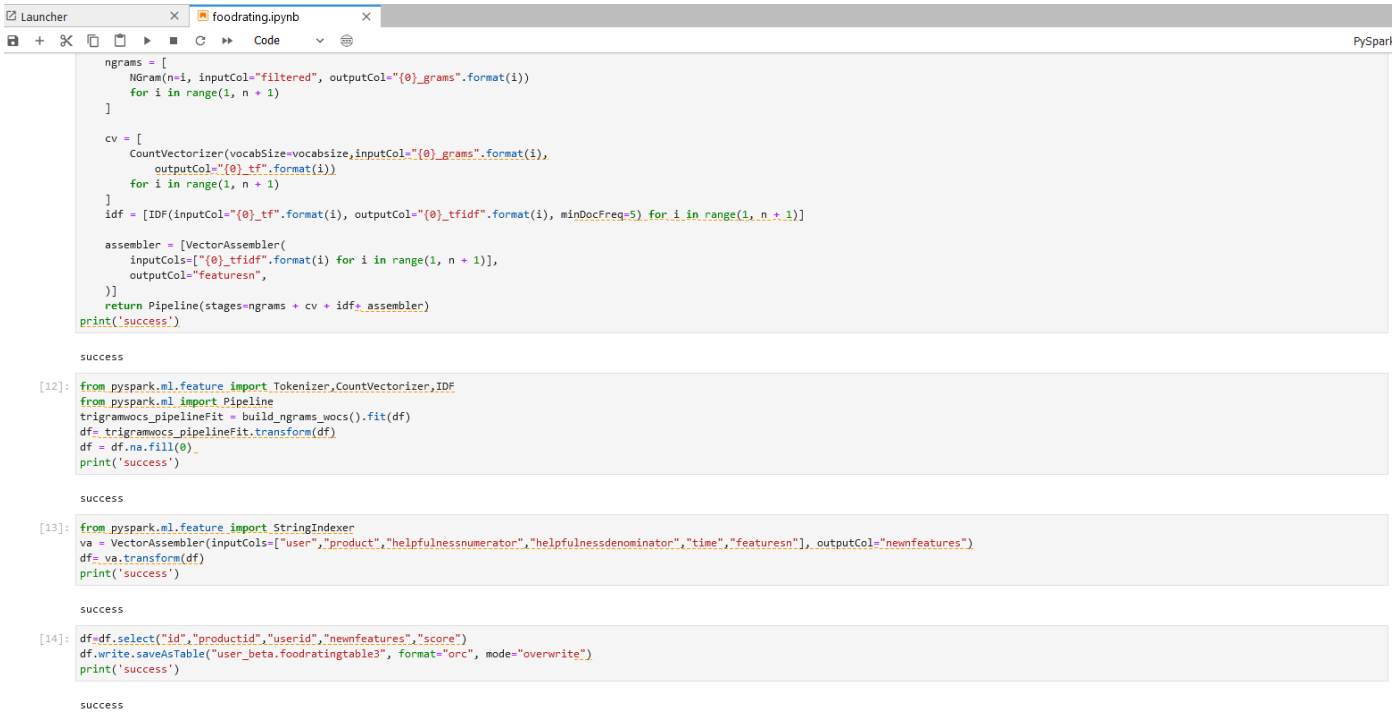
```
df=df.select("id","productid","userid","newnfeatures","score")
df.write.saveAsTable("xxx_xxx.table3", format="orc", mode="overwrite")
print('success')
```

步骤四说明 - 结果数据存储

> 将特征提取完毕的数据存入目标 Hive表中。

具体操作和步骤可以参考下图。





步骤七：线性回归模型预测

继续上一步的PySpark notebook，在PySpark程序中针对线性回归模型模型，测试不同的参数组合，最后导出最优参数组合的模型以及该模型的预测结果。

> 进入jupyterlab本项目选的PySpark环境。

```
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.替换table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.替换table为实际Hive目标表名

```
df=spark.sql("select * from xxx_xxx.table3")
(train, test) = df.randomSplit([0.8, 0.2],seed = 11)
print('success')
步骤一说明 - 数据转换与导入
```

> 导入上一个步骤的输出到Jupyter，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.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator
lr = LinearRegression(featuresCol = 'newnfeatures', labelCol='score')
lr_evaluator = RegressionEvaluator(predictionCol="prediction", \
                 labelCol="score",metricName="r2")
param_grid = ParamGridBuilder() \
             .addGrid(lr.regParam, [ .01, .05, .1, .15]) \
             .addGrid(lr.elasticNetParam, [ 0.0, 0.5, 1.0]) \
             .build()
tvs = TrainValidationSplit(estimator=lr,
                           estimatorParamMaps=param_grid,
                           evaluator=lr_evaluator,
                           trainRatio = 0.8)
model=tvs.fit(train)
lr_prediction=model.transform(test)
print('success')
lr_evaluator = RegressionEvaluator(predictionCol="prediction", \
                 labelCol="score",metricName="r2")
print("R Squared (R2) on test data = %g" % lr_evaluator.evaluate(lr_prediction))
lr_prediction.show(3)
步骤二说明 - 线性回归模型的调参与预测
```

> 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，对12种参数组合模型进行测试。

> 使用最优检测模型对数据进行拟合并输出最佳预测结果。

步骤三操作

> 注：机构项目时： 1."hdfs:///xxx/xxx/data/mllib/gbtcmodel" 为`hdfs:///org/xxx/data/mllib/lrmodel`，xxx替换为当前机构名。 2."xxx_xxx.table4"为org_xxx，xxx替换为当前机构名，table4替换为用户自定义的Hive表名。

> 个人项目时 1."hdfs:///xxx/xxx/data/mllib/lrmodel" 为`hdfs:///user/xxx/data/mllib/lrmodel`，xxx替换为当前登录用户名。 2."xxx_xxx.table4"为user_xxx，xxx替换为当前登录用户名，table4替换为用户自定义的Hive表名。

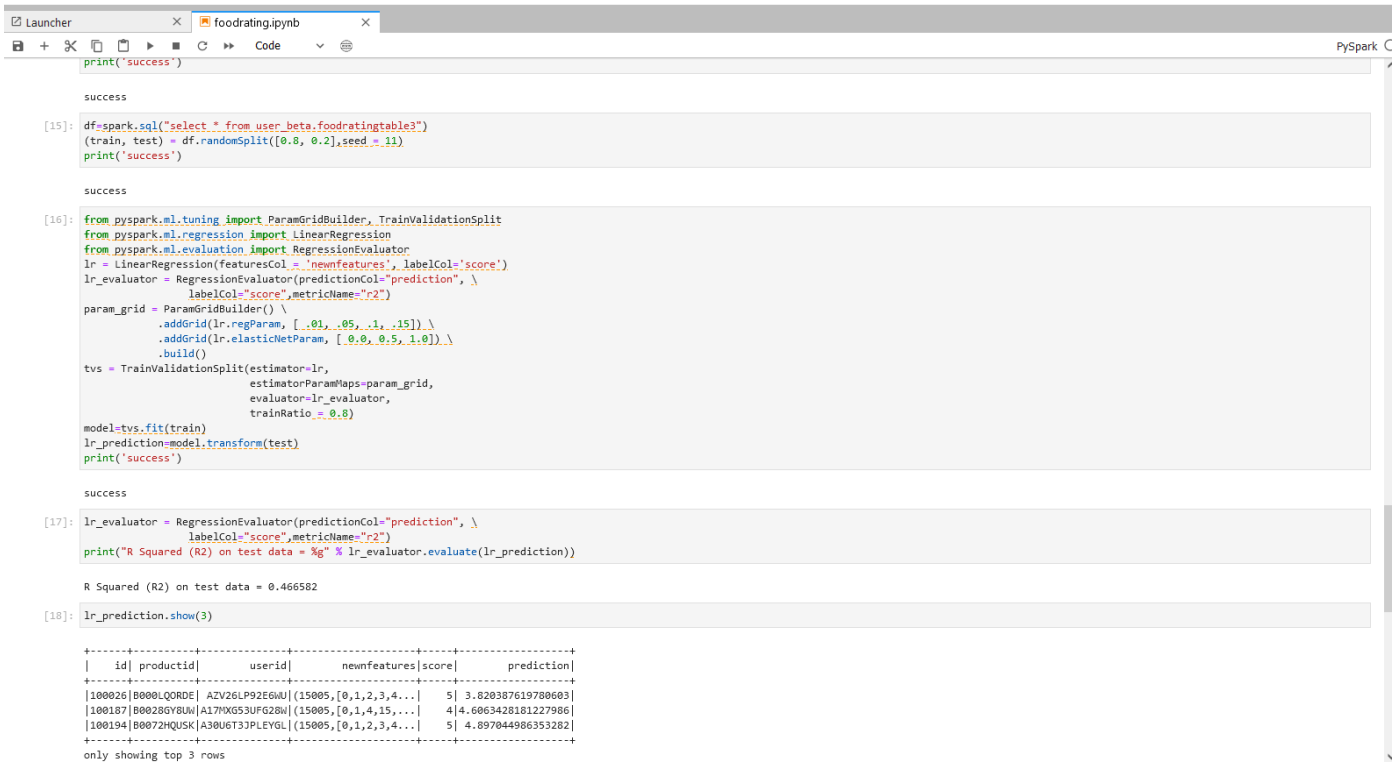
```
model.write().overwrite().save("hdfs:///xxx/xxx/data/mllib/lrmodel")
lr_prediction.write.saveAsTable("xxx_xxx.table4", format="orc", mode="overwrite")
print('success')
spark.stop()
```

步骤三说明 – 结果数据存储

> 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目标 Hive表中。

具体操作和结果可以参考下图。



Launcher | foodrating.ipynb

Code | PySpark

```
print('success')
```

success

[17]:
```
lr_evaluator = RegressionEvaluator(predictionCol="prediction", \
                 labelCol="score",metricName="r2")
print("R Squared (R2) on test data = %g" % lr_evaluator.evaluate(lr_prediction))
```

R Squared (R2) on test data = 0.466582

[18]:
```
lr_prediction.show(3)
```

```
+------+----------+--------------+--------------------+-----+------------------+
|    id| productid|        userid|        newnfeatures|score|        prediction|
+------+----------+--------------+--------------------+-----+------------------+
|100026|B000LQORDE| AZV26LP92E6WU|(15005,[0,1,2,3,4...|    5| 3.820387619780603|
|100187|B0028GY8UW|A17MXG53UFG28W|(15005,[0,1,4,15,...|    4|4.6063428181227986|
|100194|B0072HQUSK|A30U6T3JPLEYGL|(15005,[0,1,2,3,4...|    5| 4.897044986353282|
+------+----------+--------------+--------------------+-----+------------------+
only showing top 3 rows
```

[20]:
```
model.write().overwrite().save("hdfs:///user/beta/data/mllib/lrmodel")
lr_prediction.write.saveAsTable("user_beta.foodratingtable4", format="orc", mode="overwrite")
print('success')
```

success

[ ]:

[ ]: