# 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:<u>foodreview</u>。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、

Poos     Doline     数据工程     数据近程     数据资产     集成工具	政策讨论工具 管理 权用配置	devoj	os • 🗳 用户手册 🙁 beta
数項工程 数通他用 BDOS Online 新手数程・)	①         机器学习         流水线分析           自建新项目               ・             ・	最近打开 机器学习-反教许 个人项目 2021-10-11110558 beta	列表形式: 日 6 :
項目発型     第四月     ● 正常項目     ● 日間項目         ● 日間項目	B序 <sup>™</sup> □ 只看我的 □ 保留施选编 2021-11-03 114851   brta : ,	好 数据版水线 场景 (小人用用 2021-11-02 1630050 beta <u> 全址数合、个人体验 (小人用用</u> 2021-11-02 1536646 beta	: :
	2021-11-02 16:42:36 ) beta : . 特況始め「音教展与生产教展进行深邃的关联、得到限速数展模型、并从「音点击车、获高成本等 2021-11-02 15:51:34 ) beta : 2021-11-02 15:51:34 ) beta :	<u>企业数合_场票体验</u> 2021-11-02 11.03:00 beta 数据版水线_场票 个人项目	:
企业数仓_场展体验_20111102 个人原目 本项目展示了一个代型的英联网的大数量分析实例,对使生数要进行采载,处计	2021-11-02 13:49:18 : beta : 2021-11-02 13:49:18 : beta : . 分析及其例/得出等护管、旨在荷动用户快速体验大数强分析的数据完终与转换。	2021-11-02 11:03:02 beta	
	2021-11-02 11:34:18 beta : 2021-11-02 11:35:0 beta : 2021-11-02 11:35:0 beta :	v Ø devor	os ◆ □用户手册 💊 beta
Comme     Analie Soulov Manyalio     Summer Analysis			列表形式: 日 日
XQB@#H BDOS Online 原手教理・	豫加项目 ×     和成以時期目     和成以時期目     和成以時期目     和成以時期目     和成以時期目     和成以時期目     和成以時期目     和成以時期目     和成以時期目     《供留微选编》     《書書》     》     《書書》     《書書》     》     《書書》     》     《書書》     》     《書書》     《書書》     》	<ul> <li>最近打开</li> <li>机器学习之关会(16:08)</li> <li>2021-05-10-03-45:32</li> <li>bela</li> <li>和器学习-NBA音(小和目 2021-11-0914-58:52</li> <li>bela</li> </ul>	
2 11040203 (小規制) 利24時日期1409(何約5日)	*项目名称 机晶学习-美食评分预测-个人体组 72339 beta : 项目描述	NBA赛手比查预测 ①人用目 2021-06-23 16-32:07 em 机器学习-电影推荐 ①人用目 2021-09-16 09:17:28 beta	
机器学习-NBA赛季比查预测-个人体验 《人原目	ROIR ROIR ASES2   beta :	<b>机器学习-电影推 个人原用</b> 2021-11-08 14:34:21 beta	
机器学习-电影推荐-个人体验(个人原则)	2021-11-08 1434-21   beta 🕴		
	2021-11-06 12:02:39   beta : 2021-11-05 12:05:09   beta : 2021-11-05 11:05:09   beta :		

### 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。

<b>P</b> online	数据工程 集成工具 数据浏览	工具 管理 权限起置	devops *	□ 用户手册	B beta
当前项目步骤 () 《 点击切换步骤洋情	数据采集 * 数据转换 * 数据	3分析 * 数据质量 * 数据版务 * 数据应用 * Bi根表 *			
1-URL文件导入 URL文件导入	数据工程 / 个人项目 / 机器学习-美	食评分预测·个人体验 URL文件导入 1-URL文件导入 ▲			
2-HDFS到Hive导入 HDFS웨Hive导入	步骤配置 运行记录 步骤R	EADME 教理 留貴版			
3-JupyterNotebook	过程编写				导入
	导入数据源配置	* 目标URL下载			
<b>(</b>	配置需要导入的目标URL和文件名	http://linktime-public.oss-cn-qingdao.aliyuncs.com/Project_online/Jupyter/ratingsdemo2.csv			
·/		若输入的URE没有httpsighttp前圆,则合款认加上"https://作为前圆			
		文件名称			
		food ratingsdemo1109.csv			
		書画 Gan 例如:.csv, txt, xls			
	目标目录设置	HDFS目录选择			
	选择目标目录	/user/beta 点击浏览			
		不填表示默认远经当前目录 可输入"/" 查香井还提了目录			
		是否选择覆盖同名文件			

# 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。

数据采集  数据转换  数据	品分析 · 数据质量 · 数据服务 · 数据应用 · Bl服表 ·
数据工程 / 个人项目 / 机器学习-3	έ食评分预测一个人体验 HDFS到Hive导入 2-HDFS到Hive导入 ∠
	LADME 教程 留高版
HDFS数据源配置	HDFS目标文件选择
文件采集,从HDFS加载到数据库	/user/beta 点击浏览
	只能选择当前目录或者子目录下的文件,可输入"/"宣告并选择文件列表
目标数据库类型: Hive	較調库 user beta ····································
	Nutre ++ x
	<ul> <li>● 表已存在 ○ 新表示创建</li> </ul>
	数据表
	方式
	追加 ~
	导入并保存

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。

對對数	据表导入向导							×	×
1-U URL:			⊘ 文件格式				● 导入数据库		
2-H HDF 3-JI	目标表名称	user_ 表名称格:	beta.food_ratingsdemo1109 式为数据语表,只能使用大小写字母以及下划线和数字。例如 us	er_test.table1				^	
Jupy	数据格式 ●	CSV							
<sup>44</sup> 如数据中 <sup>Jur</sup> 建议选择 5-Jt	中包含双引号、时间 Ecsy的数据格式	■電導特殊 ■ 使用	字符时. ]算——行作为表头						
Jupy	字段	名称中只的	能包含大小写字母,数字以及下划线						
		名称	ld	类型	bigint	1	2		
		名称	ProductId	类型	string	B001E4KFG0	B00813GRG4		
		名称	UserId	类型	string	A3SGXH7AUHU8GW	A1D87F6ZCVE5NK		
		名称	ProfileName	类型	string	delmartian	dll pa		
		名称	HelpfulnessNumerator	类型	boolean	1	0		
		名称	HelpfulnessDenominator	类型	boolean	1	0	_	1
		名称	Score	类型	bigint	5	1		
		名称	Time	类型	bigint	1303862400	1346976000		,
								上一步 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。

8 +	ж	C □ □ ► ■ C → Code < ⊕	PySpark
(	[6]:	<pre>data-spark.sql("select * from user_beta.food_ratingsdemo1110") data-data.filter(data.text!="Text") data.show(1)</pre>	
		id  productid  userid  profilename helpfulnessnumerator helpfulnessdenominator score  time  summary  text    1 8001E4KF60 A3SGXH7AUAHU8GW  delmartian  1  1  5 1303862400 Good Quality Dog I have bought sev	
[	[7]:	<pre>data=data.distinct() data=data.dropDuplicates(subset=[c for c in data.columns if c in ["productid", "userid", "time"]]) data.count()</pre>	
		170405	
[	[8]:	<pre>import pyspark.sql.functions as f data.agg("[[1-(f.count(c) 'f.count(c) 'f.count(c) 'i.song') for c in data.columns]).show() data-data.dropna(thresh=3) data-data.dropna(thresh=3)</pre>	
		++  id missing productid missing userid missing profilename missing helpfulnessnumerator missing helpfulnessdenominator missing score missing time missing summary missing text missing	
		······································	
[	[9]:	<pre>data = data.withColumn("helpfulnessnumerator", data['helpfulnessnumerator'].cast('int')) data = data.withColumn("helpfulnessdenominator", data['helpfulnessdenominator'].cast('int')) data = data.withColumn("time", data['time'].cast('int')) print('success')</pre>	
		success	
[1	10]:	<pre>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))</pre>	
		++  score count   ++	
1			
🗷 foodr	ating	g1.ipynb X	
8 +			
E 4	×		PySpark (
[1	<b>%</b> 0]:	Image: Control of the second secon	PySpark (
[1	¥ .0]:	Image: Control of the second secon	PySpark (
[1	<b>X</b>	<pre>Image: Image: Imag</pre>	PySpark (
[1	¥ .0]:	<pre>Image: Image: Imag</pre>	PySpark (
[1	¥ .0]:	<pre>Image: Image: Imag</pre>	PySpark (
[1	×	<pre>     C → Code      Code</pre>	PySpark (
[1	×	<pre>     C → Code 、      Code      Code 、      Code 、      Code      Code 、      Code 、</pre>	PySpark (
[1	X 0]:	<pre>     C → Code 、      Code      Code 、      Code 、      Code 、      Code</pre>	PySpark (
[1	X 0]:	<pre>     C → Code 、      Code     Code 、      Code 、      Code 、      Code</pre>	PySpark (
[1	X 0]:	<pre>     C → Code 、      C → Code 、      Code 、      C → Code      C → Code 、      C → Code      C → Code</pre>	PySpark (
[1	× .0]:	<pre>     C → Code → C</pre>	PySpark (
[1	X 0]:	Image: Design of the second secon	PySpark (
[1	× 00]:	Image: Comparison of the control of	PySpark (
[1	× 00]:	Image: C + Code       Image: C + Code         from pyppark.sal.functions import.coll         data_souppy("score").count().orderBy(col("count").desc()).show(truncate-False)         data_sdata.filter(data.score = 0)[data.score = 2).[data.score = 3).[data.score = 4][data.score = 5))         score[count]         iscore[count]	PySpark (
[1	0]: X	Image: Description: Select: Cold         data_group(core=).count().onder=0(count*).desc()).show(truncte=False)         data_data_data.tilter((data.score == 0)](data.score == 2).l (data.score == 3).l (data.score == 5))	PySpark (
(1	<b>X</b> ∞]:	<pre>     C</pre>	PySpark (
[1	<b>X</b> ∞]:	<pre>     C</pre>	PySpark (
[1	<b>X</b> .0]: □1]: □	<pre>     C</pre>	PySpark (
[1 [1	<b>X</b> .0]: .1]: .2]:	<pre>     Compary Act Act Act Act Act Act Act Act Act Act</pre>	PySpark (
(1 (1	X (0): (1): (2): (2):	<pre>     C</pre>	PySpark (
[1 [1]	<b>X</b> 00]: .1]: .2]:	<pre>     C</pre>	PySpark (

+ %	T T + C + Code - Code
[7]:	<pre>from pyspark.sql.functions import regexp_replace data = data.withColumn("only_str",regexp_replace(col('summary'), '\d+', '')) print('success')</pre>
	success
[8]:	<pre>from pyspark.ml.feature importRegexTokenizer_StopWordsRemover regex_tokenizer = RegexTokenizer(inputCol="only_str", outputCol="mwords", pattern="\\W") data = regex_tokenizer_transform(data) removerStopWordsRemover(inputCol="mwords", outputCol="filtered") data = remover_transform(data) data.select("mwords","filtered").show()</pre>
	<pre>mwords filtered [love, the, book,[love, book, miss] [canine, crack] [canine, crack] [lots, of, crispy[lots, crispy, w] [i, love, trispy[lots, crispy, w] [i, love, this, m] [love, movie] [fleas, sre, st] [flea, stuck]] [needs, improved] [needs, improved] [iexcl, oye, trem][excl, oye, trem] [read, the, fine,] (read, fine, print]] [as, good, as, we] [good, barcelona] [desen, t, taste] (desen, taste, li] [pleasantly, surp][pleasantly, surp] [a, disappointed,] (disappointed, bo] [excellent, produ] (excellent, produ] [really, hot, but] [really, hot, rea] [lonst.the, right,] [right, bottle, s] [omaha, cheesceak] [omaha, cheesceak] [slightly, remin] [slightly, remin] [mexican, mocha] [mexican, mocha] </pre>
[9]:	<pre>data.write.format("hive").mode("overwrite").saveAsTable("user_beta.foodratingtable?") print('success')</pre>

## 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。

PySpark

进入jupyterlab本项目选的PySpark环境

### 步骤一操作

注:

۲

```
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)
    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。

🖾 Launche	ncher 🛛 📉 🗖 foodrating.ipynb 🛛 🕹	
8 + 3	- ‰ [î] ▶ ■ C → Code ∨ ⊕	PySpark (
[9]	[9]: data.write.format("hive").mode("overwrite").saveAsTable("user_beta.foodr	itingtable2")
	<pre>print('success')</pre>	
	SUCCESS	
[10]	<pre>[10]: df<u>=spark.sql("select * from user_beta.foodratingtable2")</u> print('success')</pre>	
	success	
[11]	[11]: vocabsize=5000	
	from pyspark.ml.feature import NGram, VectorAssembler	
	ngrams = [	
	<pre>NGram(n=i, inputCol="filtered", outputCol="{0}_grams".format(i)) for i in press(1, p + 1)</pre>	
	]	
	ev = [	
	CountVectorizer(vocabSize=vocabsize,inputCol="{0} grams".format(	
	<pre>outputCol="{0} tf".format(i)) for i in carge(1 n + 1)</pre>	
	]	
	<pre>idf = [IDF(inputCol="{0}_tf".format(i), outputCol="{0}_tfidf".format</pre>	i), minDocFreg=5) for i in range(1, n + 1)]
	assembler = [VectorAssembler(	
	<pre>inputCols=["{0}_tfidf".format(i) for i in range(1, n + 1)], outputCol="featuresn"</pre>	
	)]	
	<pre>return Pipeline(stages=ngrams + cv + idf<u>+_assembler)</u> nrint('success')</pre>	
	print( success )	
	success	
[12]	[12]: from pyspark.ml.feature import Tokenizer,CountVectorizer,IDF	
	from pyspark.ml import Pipeline	
	<pre>trigramwocs_pipelineFit = build_ngrams_wocs().fit(df) dftrigramwocs_pipelineFit.transform(df)</pre>	
	df = df.na.fill(0)	
	print('success')	
	success	
[13]	[13]: from nyspack.ml.feature import StringIndexer	
[	[10], Haw by share and the share of a Bruncher	
	<pre>va = VectorAssembler(inputCols=["user","product","helpfulnessnumerator",</pre>	<pre>'helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures")</pre>
	<pre>va = VectorAssembler(inputCols=["user","product","helpfulnessnumerator", dfva.transform(df)</pre>	helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures")
2 Launcher	va = VectorAssembler(inputCols=["user","product","helpfulnessnumerator", df=_va.transform(df) cher X	helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures")
☑ Launcher	va = VectorAssembler(inputCols=["user","product","helpfulnessnumerator.",       df=_va.transform(df)	helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures")
☑ Launcher ₽ + 3	<pre>va = VectorAssembler(inputCols=["user","product","helpfulnessnumerator", dfe_va.transform(df) cher</pre>	helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures") PySpar
☑ Launcher ■ + 영	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", dfva.transform(df) cher</pre>	"helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures") PySpar
🖾 Launchei	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df=_va.transform(df) cher X</pre>	"helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures"). PySpar
☑ Launchei ■ + 3	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df=_va.transform(df)  cher</pre>	helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures"). PySpari
☑ Launcher B + 3	<pre>va = VectorAssembler(inputCols=["user", "product", "helpfulnessnumerator", df=_va.transform(df)  cher X</pre>	helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures"). PySpar
☑ Launcher	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df=_va.transform(df)  cher</pre>	<pre>helpfulnessdenominator"."time","featuresn"), outputCol="newnfeatures") PySpar </pre>
☑ Launcher ■ + 8	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df_va.transfore(df)  cher</pre>	<pre>helpfulnessdenominator"."time","featuresn"). PySpar })</pre>
☑ Launcher ■ + 8	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df_va.transform(df)  cher</pre>	<pre>belpfulnessdenominator"."time","featuresn"), outputCol="newnfeatures")  PySpar  i),  (1), minDocfreg-5)_for_i_in_range(1, n, + 1)]</pre>
☑ Launcher ■ + å	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", dfva.transform(df)  cher</pre>	<pre>belpfulnessdenominator"."time","featuresn"), outputCol="newnfeatures")  PySpar  i), (i), minDocFreg-5)_for_i_in_range(ln_+_1)]</pre>
I <sup>2</sup> Launcher ■ + 8	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df=_va.transform(df)  cher</pre>	<pre>belpfulnessdenominator"."time","featuresn").  PySpan  i),  (i), minDocFreg=5) for i in range(1, n, + 1)]</pre>
⊠ Launcher a + 3	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df=_va.transform(df) kter X</pre>	<pre>belpfulnessdenominator","time","featuresn"), outputCol="newnfeatures")  PySpan  i),  (i), minDocFreq=5) for i in range(1, n + 1)]</pre>
년 Launcher	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df_wa.transform(df)  cher</pre>	<pre>helpfulnessdenominator"."time","featuresn").  PySpar  i),  (i), minDocFreg-5).for.i.in.range(1, n.+.1)]</pre>
☑ Launchen ₽ + 8	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df_wa.transform(df) cher</pre>	<pre>helpfulnessdenominator"."time","featuresn")  PySpar  i),  (i), minDocFreq=5) for i in range(1, n + 1)]</pre>
☑ Launcher	<pre>va = VectorAssembler(inputCols=["user", "product", "helpfulnessnumerator", df_va.transform(df)  cher</pre>	<pre>helpfulnessdenominator"."time","features")  PySpar  i),  (1), minDocfreg=5)_for_i_in_range(1, n, r, 1)]</pre>
☐ Launchen ■ + 8	<pre>va = VectorAssembler(inputCols=["user", "product", "helpfulnessnumerator", df_va.transform(df)  cher</pre>	<pre>helpfulnessdenominator"."time","features")  PySpar  i), (i), minDocfreg-5)_for_i_in_range(1, n, t, 1)]</pre>
Launchen 1 + 3	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df_va.transform(df) kcher X Production Production Product Pro</pre>	<pre>helpfulnessdenominator"."time","features")  PySpar  i), (i), minDocfreq-5)_for_i_in_range(1, n, t, 1)]</pre>
[2] Launchen ■ + 8 [12]	<pre>va = VectorAssembler(inputCols=["user", "product", "helpfulnessnumerator", df_wa.transform(df)  cher</pre>	<pre>helpfulnessdenominator"."time","featuresn").  PySpar  i), (i), minDocfreg-5)_for_i_in_range(l_k_n_+_1)] </pre>
[2] Launchee P + 3 [12]	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df_wa.transform(df) cher</pre>	<pre>helpfulnessdenominator"."time","features")  PySpar  i),  (i), minDocFreg=5) for_i_in_range(1, n + 1)] </pre>
[2] Launcher Part + 8 [12]	<pre>va = VectorAssembler(inputCols=["user", "product", "helpfulnessnumerator", df_wa.transform(df) cher</pre>	<pre>helpfulnessdenominator"."time","features")  PySpar  i), (i), minDocFreq=5) for i in range(1, n + 1)]</pre>
[2] Launcher Part + 8 [12]	<pre>va = VectorAssembler(inputCols=["user", "product", "helpfulnessnumerator", df_wa.transform(df) cher</pre>	<pre>helpfulnessdenominator"."time","features")  PySpar  i), (i), minDocfreq=5) for i in range(1, n + 1)]</pre>
[2] Launchen Pa + 8 [12]	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df_va.transform(df) where va.transform(df)</pre>	<pre>helpfulnessdenominator"."time","features")  PySpar  i),  (1), minDocfreq-5) for i in nange(1, n,t 1)] </pre>
[12]	<pre>va = VectorAssembler(inputCols=["user", product", "helpfulnessnumerator", df_va.transform(df) kter</pre>	<pre>helpfulnessdenominator"."time"."featuresn"). outputCol="newnfeatures")  i),  (i), minDocfreg-5) for i in range(1, n, t 1)]  helpfulnessdenominator" "time" "featuresn", outputfol="newnfeatures")</pre>
[2] Launchee <b>a</b> + 3 [12]	<pre>va = VectorAssembler(inputCols=["user", "product", "helpfulnessnumerator", dr. va.transform(df)</pre>	<pre>helpfulnessdenosinator", "time", "featuresn"), outputCol="newnfeatures")  i),  i),  i),  minDocFreg-5) for i in range(1, n + 1)]  helpfulnessdenosinator", "time", "featuresn"), outputCol="newnfeatures")</pre>
[2] Launchen m + 3 [12] [13]	<pre>va = VectorAssembler(inputCols=["user","product","helpfulnessnumerator", df_wa.transform(df) cher</pre>	<pre>helpfulnessdenominator"."time","featuresn"], outputCol="newnfeatures")  pySpar  j), (i), minDocfreg=5) for i in range(l. n. t. 1)]  helpfulnessdenominator"."time", "featuresn"], outputCol="newnfeatures")</pre>
[2] Launcher m + 8 [12] [13]	<pre>va = VectorAssembler(inputCols=["user","product","helpfulnessnumerator", df_wa.transform(df) kter</pre>	<pre>helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures")  i),  i),  i), minDocfreg=5) for_i in_range(1, n. r. 1)]  helpfulnessdenominator", "time", "featuresn"], outputCol="newnfeatures")</pre>
[12]	<pre>va = VectorAssembler(inputCols=["user", "product", "helpfulnessnumerator", df_wa.transform(df)  cher</pre>	<pre>helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures")  pySpar  j, (i), minDocfreg=5).for_1.in_range(1, n.+.1)]  helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures") </pre>
[12]	<pre>va = VectorAssembler(inputCols=["user","product","helpfulnessnumerator", df_va.transform(df) kter</pre>	<pre>helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures")  pySpar  i),  (i), minOpocfreg=5)_for_i_im_range(ix_0,*,1)]  "helpfulnessdenominator","time","featuresn"], outputCol="newnfeatures")  remrite")</pre>
[12]	<pre>va = VectorAssembler(inputCols=["user","product","helpfulnessnumerator", df_va.transform(df) kter</pre>	<pre>helpfulnessdenominator","time","featuresn"). outputCol="newnfeatures")  i),  (i), minDocfreg=5).for_i_in_range(1, n.t.1)]  helpfulnessdenominator","time","featuresn"). outputCol="newnfeatures")  remuritg").</pre>

## 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 ×	
<b>⊡</b> + %	Ĩ Î ▶ III C → Code - < ⊜	ySpark C
	print('success')	
	success	
[15]:	<pre>df=spark.sql("select "_from user_beta.foodratingtable3") (train, test) = df.randomSplit([0.8, 0.2]_seed11) print('success')</pre>	
	success	
[16]:	<pre>from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit from pyspark.ml.regression import Regression(import LinearRegression(featuresci), import Regression(featuresci), import Regression(featuresci),</pre>	
	success	
[17]:	<pre>lr_evaluator = RegressionEvaluator(predictionCol="prediction", \</pre>	
	R Squared (R2) on test data = 0.466582	
[18]:	<pre>lr_prediction.show(3)</pre>	
	id  productid  userid  newmfeatures score  prediction	
	1000260 [0000LQORDE]. AZVZGLP92EGAU[(15005, [0,1,2,3,4]       5 [ 3.8203876137980603]         100137 [000257Wu][A17WX055WUG][A17WX055WUG][A17WX057WG6]       [ 3.0203876372086]         100194 [00072HQUSK [A30UGT37PLEYGL] (15005, [0,1,2,3,4]       5 [ 4.897044986353282]	
	only showing top 3 rows	

🖾 Laun	cher	X 🗷 foodrating.ipynb X	
+	Ж	「□□□ ▶ ■ C → Code ~ 曲	-ySpark
		princ(success)	
[	17]:	<pre>lr_evaluator = RegressionEvaluator(predictionCol="prediction", \</pre>	
		R Squared (R2) on test data = 0.466582	
[	18]:	<pre>lr_prediction.show(3)</pre>	
		id productid userid newnfeatures score prediction	
		100026         B000LQQRDE         AZV26LP92E6WU         (15005,[0,1,2,3,4         5]         3.820387619780603           100187         B0028GY8UW         AI7MXG53UFG28W         (15005,[0,1,4,15,]         4 4.6663428181227986            100194         B0072HQUSK         A30UGT3JPLEYGL         (15005,[0,1,2,3,4]         5          4.897044386353282	
		only showing top 3 rows	
[	20]:	<pre>model.write().save("hdfs:///user/beta/data/mllib/Irmodel")_ lr_prediction.write.saveAsTable("user_beta.foodratingtable4", format="orc", mod<u>e="overwrite")</u> print('success')</pre>	
		success	
	: 1:		
	: ]:		