

# 基于 Spark 做一个歌手推荐系统

## 0. Pysaprk 安装

步骤:

### 【工具准备】

jdk1.8, spark-2.3.2-bin-dadoop2.7, hadoop-2.7.4.tar, winutils

具体如下:

1. Java: JDK 8u121 with NetBeans 8.2, 你也可以下载单独的 JDK 8u121, 不带有 IDE NetBeans 8.2

<http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html>

2.spark: spark-2.2.0-bin-hadoop2.7

<https://spark.apache.org/downloads.html>

3.winutils.exe: 用于改变文件或文件夹读写权限的工具, 本文下载的是针对 Hadoop-2.7.4 的 64 位的 winutils.exe

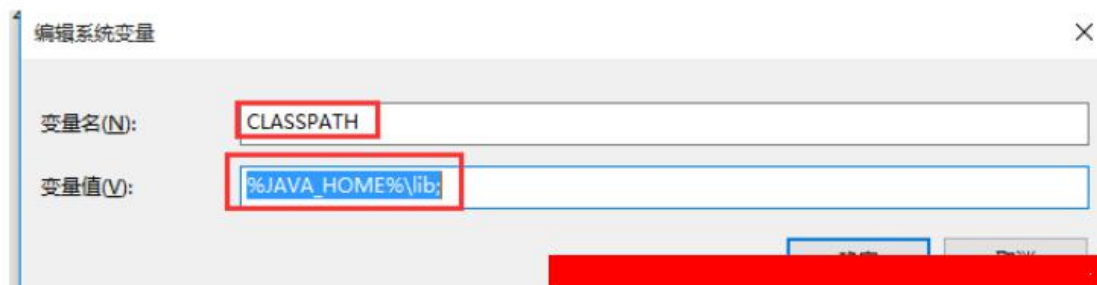
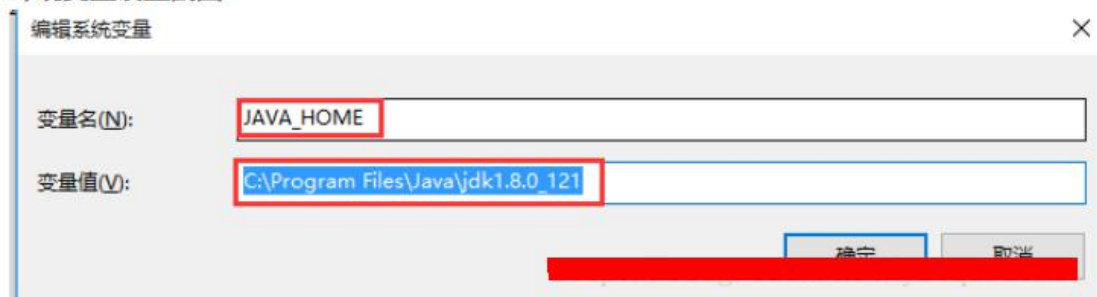
<https://github.com/steveloughran/winutils/tree/master/hadoop-2.6.4/bin>

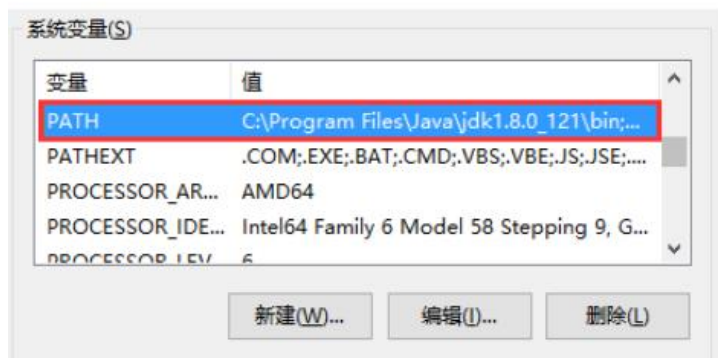
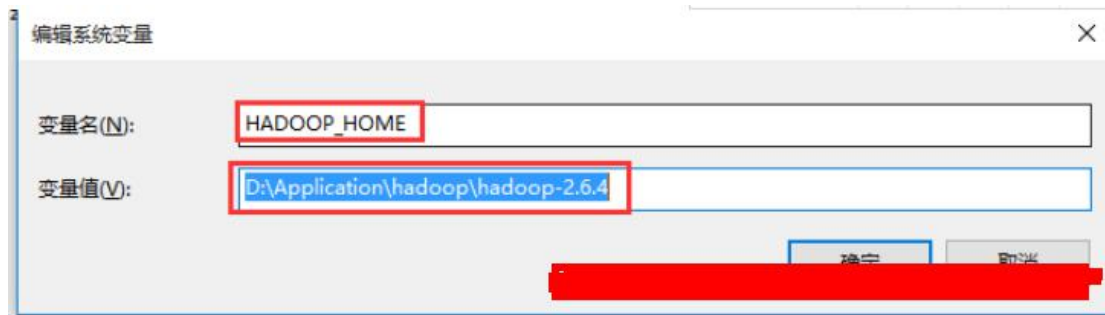
4.hadoop-2.7.4 可到官网自行下载, 或是从下文中分享文件中获得

### 【安装及系统环境变量设置】

- 安装 Jdk 的安装。
- Hadoop 和 Spark 解压就可以, 但是注意的是要确保解压的路径中不包含空格!
- 然后是配置环境变量

环境变量设置截图





注意：编辑系统变量 PATH 的值，将 java,hadoop 的相关 bin 路径添加进去，spark 添加到它的主目录就够了，注意是添加，不要删除原有的其他应用的路径值！！！每个路径之间用英文半角的引号(;)分开

## 【将 pyspark 文件放到 python 文件夹下、使用 winutils.exe 修改权限】

1，将 spark 所在目录下（D:\Software\spark-2.3.2-bin-hadoop2.7\python）的 pyspark 文件夹拷贝到 python 文件夹下（D:\Program Files\anaconda\Lib\site-packages）

具体目录要看大家自己安装的时候是放在哪的！

2，安装 py4j 库

一般的在 cmd 命令行下 `pip install py4j` 就可以。若是没有将 pip 路径添加到 path 中，就将路径切换到 python 的 Scripts 中，然后再 `pip install py4j` 来安装库。

3，修改权限

将 winutils.exe 文件放到 Hadoop 的 bin 目录下（D:\Software\hadoop-2.7.3\bin），然后以管理员的身份打开 cmd，然后通过 `cd`

命令进入到 Hadoop 的 bin 目录下，然后执行以下命令：

```
winutils.exe chmod 777 c:\tmp\Hive
```

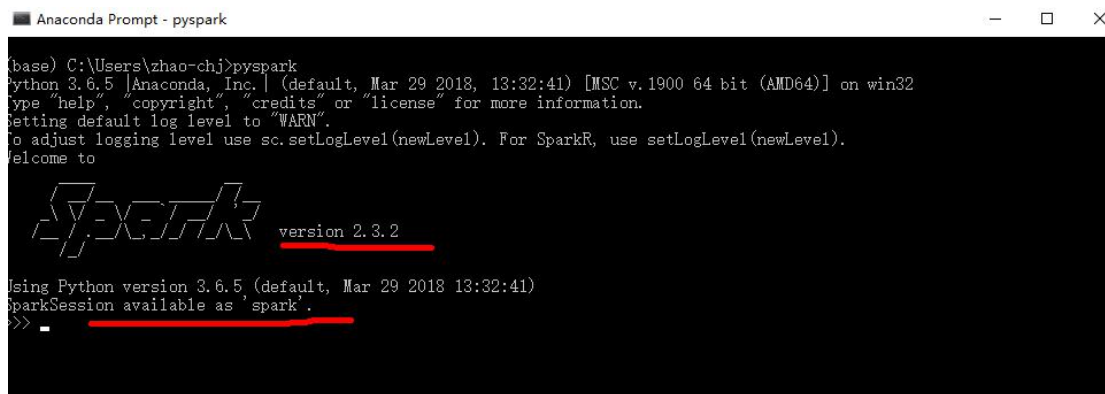
**注意：**

1，cmd 一定要在管理员模式下！cmd 一定要在管理员模式下！cmd 一定要在管理员模式下！

2，‘C:\tmp\hive’，一般按照上面步骤进行了之后会自动创建的，一般是在 Hadoop 的安装目录下出现。但是若没有也不用担心，自己在 c 盘下创建一个也行。

## 【验证安装成功】

关闭命令行窗口，重新打开命令行窗口，输入命令：pyspark



```

(base) C:\Users\zhao-chj>pyspark
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

  ____      __
 / ___ |    /  |
/ /___|    /   |
 \___  |   /    |
  ___| |  /_____|
  |___|_|

version 2.3.2

Using Python version 3.6.5, (default, Mar 29 2018 13:32:41)
SparkSession available as 'spark'.
>>> 
```

出现上面的窗口且不报错，那么恭喜你安装成功了！

## 问题记录 1：

Exception: Java gateway process exited before sending the driver its port number

**解决办法：**

[https://blog.csdn.net/yhp\\_daniel/article/details/82468637](https://blog.csdn.net/yhp_daniel/article/details/82468637)

## 问题记录 2：

版本设置问题：

<https://www.jianshu.com/p/f345e86971e5> jupyter 版本设置

<https://stackoverflow.com/questions/44270341/spark-execution-error-python-in-worker-has-different-version>

<https://blog.csdn.net/moledyzhang/article/details/78856467>

<https://blog.csdn.net/HHTNAN/article/details/80284139> 报错集锦:

### 问题记录 3:

```
(base) C:\Users\zhao-chj>pyspark
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
Traceback (most recent call last):
  File "D:\ProgramCJ\spark-2.0.2-bin-hadoop2.7\bin\..\python\pyspark\shell.py", line 30, in <module>
    import pyspark
  File "D:\ProgramCJ\spark-2.0.2-bin-hadoop2.7\python\pyspark\__init__.py", line 44, in <module>
    from pyspark.context import SparkContext
  File "D:\ProgramCJ\spark-2.0.2-bin-hadoop2.7\python\pyspark\context.py", line 33, in <module>
    from pyspark.java_gateway import launch_gateway
  File "D:\ProgramCJ\spark-2.0.2-bin-hadoop2.7\python\pyspark\java_gateway.py", line 31, in <module>
    from py4j.java_gateway import java_import, JavaGateway, GatewayClient
  File "<frozen importlib._bootstrap>", line 971, in _find_and_load
  File "<frozen importlib._bootstrap>", line 955, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 656, in _load_unlocked
  File "<frozen importlib._bootstrap>", line 626, in _load_backward_compatible
  File "D:\ProgramCJ\spark-2.0.2-bin-hadoop2.7\python\lib\py4j-0.10.3-src.zip\py4j\java_gateway.py", line 18, in <module>
    import inspect
  File "D:\ProgramCJ\Anaconda\Anaconda3\lib\inspect.py", line 361, in <module>
    Attribute = namedtuple('Attribute', 'name kind defining_class object')
  File "D:\ProgramCJ\spark-2.0.2-bin-hadoop2.7\python\pyspark\serializers.py", line 381, in namedtuple
    cls = _old_namedtuple(*args, **kwargs)
TypeError: namedtuple() missing 3 required keyword-only arguments: 'verbose', 'rename', and 'module'
'''
```

### 【解决办法】

pyspark 的使用过程中，spark 版本小于 2.1 的是不支持 python3.6 版本的，所以将 python3.6 的版本更换成小于 3.6 的版本。至此，问题解决

```
Anaconda Prompt - pyspark
(base) C:\Users\zhao-chj>pyspark
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

  ____  __
 / ___/  / /  ____  ____
/ /   /  / /  / __ \/ __ \
/ /___/  / /  / ___/ / ___/
/_____/  /_/  /_/  /_/_/

version 2.3.2

Using Python version 3.6.5 (default, Mar 29 2018 13:32:41)
SparkSession available as 'spark'.
>>> _
```

# 1.代码详解

用 pandas 和 spark 做推荐系统。我们的数据源是后台收集的用户听了哪些歌手的歌曲，负责数据的同事将清洗好的歌手数据给到之后，是这个样子的：

	A	B	C
1	1523260590511gsTtu	邓丽君	2018/4/20 10:13
2	1523413580330V4Wpf	赵雷	2018/4/20 14:12
3	1522750485469mzXqM	一朵	2018/4/20 14:28
4	1524041544624GyRG	小虎队	2018/4/20 14:40
5	1521534475691jzvzf	刘德华	2018/4/20 16:15
6	1521534475691jzvzf	周杰伦	2018/4/20 16:30
7	1521534475691jzvzf	五月天	2018/4/20 16:30
8	1521534475691jzvzf	林俊杰	2018/4/20 18:38
9	1521534475691jzvzf	刘德华	2018/4/20 18:41
10	1521534475691jzvzf	赵雷	2018/4/20 20:13
11	1521534475691jzvzf	赵雷	2018/4/20 20:15
12	1521534475691jzvzf	赵雷	2018/4/20 20:16
13	1516925494173aQO5n	刘德华	2018/4/21 16:56
14	15232573243575bUSQ	周杰伦	2018/4/21 18:30
15	15242195457521H4J	林俊杰	2018/4/23 11:24
16	1523845728145zOfL	张学友	2018/4/23 11:44
17	1523413580330V4Wpf	许嵩	2018/4/23 11:55
18	1523845728145zOfL	许嵩	2018/4/23 11:56
19	1516925494173aQO5n	赵雷	2018/4/23 18:51
20	15234269125215oH4a	赵雷	2018/4/23 19:20
21	1521534475691jzvzf	赵雷	2018/4/23 19:20
22	15242195457521H4J	王力宏	2018/4/23 21:44

这是部分用户听过的歌手数据，基本也都是一些测试的数据，A 列表示我们用户的后台 ID，B 列表示用户所听的歌手，C 列表示用户听歌的时间。拿到这些数据，首先就要思考一下如何来使用这些数据，我的认为是用户听某个歌手的频次越高，就代表这个用户越喜欢这个歌手，暂且就把听某个歌手的频次作为这个用户对这个歌手的评分。首先，我们要对这个原始的数据进行预处理，这里我们运用的是 pandas。首先我们运用 pandas 读取我们的.csv 数据并且给与每一列命名：

```
singer_recom = pd.read_csv("dataset/singer.csv", header = None, encoding = 'gbk', names = ['UserTitle', 'SingerTitle', 'Date'])
```

然后我们可以看看这个列表：

```
singer_recom.head()
```

	UserTitle	SingerTitle	Date
0	1523260590511gsTtu	邓丽君	2018/4/20 10:13
1	1523413580330V4Wpf	赵雷	2018/4/20 14:12
2	1522750485469mzXqM	一朵	2018/4/20 14:28
3	1524041544624GyRG	小虎队	2018/4/20 14:40
4	1521534475691jzvzf	刘德华	2018/4/20 16:15

读取了这些数据之后，我们可以去掉重复的数据：

```
singer3 = singer_recom.drop_duplicates(subset = 'UserTitle', keep =  
'first')
```

去掉重复数据之后，我们给每一个用户 ID 分配一个 int 型的 ID，并创建一个用户和歌手的列表，方便后续的计算：

```
singer3['UserID'] = range(0, singer3.shape[0])
```

```
columns = ["UserTitle", "UserID"]
```

```
singer_user = singer3[columns]
```

我们来看看分配了数据之后的用户 ID：

```
singer_user.head()
```

	UserTitle	UserID
0	1523260590511gsTtu	0
1	1523413580330V4Wpf	1
2	1522750485469mzXqM	2
3	1524041544624GyRG	3
4	1521534475691jzvzf	4

可以看到我们给每一个用户的 **UserTitle** 都分配了一个 **UserID**，接着我们将 **user\_id** 转化成一维数组，索引对象是我们的 **user**，在这里我们用的是 **pandas** 的 **Series**：

```
user = singer_user['UserTitle'].values

user_id = singer_user['UserID'].values

series_custom = Series(user_id, index = user)
```

然后我们再将 **UserID** 添加在原有的列表中的最后一列：

```
trans_id = []

for i in singer_recom["UserTitle"]:

    trans_id.append(series_custom[i])

singer_recom["UserID"] = trans_id
```



我们可以得到新的列表，这个列表中心添加了一列 UserID:

singer_recom				
	UserTitle	SingerTitle	Date	UserID
0	1523260590511gsTtu	邓丽君	2018/4/20 10:13	0
1	1523413580330V4Wpf	赵雷	2018/4/20 14:12	1
2	1522750485469mzXqM	一朵	2018/4/20 14:28	2
3	1524041544624GyRG	小虎队	2018/4/20 14:40	3
4	1521534475691jzvzf	刘德华	2018/4/20 16:15	4
5	1521534475691jzvzf	周杰伦	2018/4/20 16:30	4
6	1521534475691jzvzf	五月天	2018/4/20 16:30	4
7	1521534475691jzvzf	林俊杰	2018/4/20 18:38	4
8	1521534475691jzvzf	林俊杰	2018/4/20 18:38	4
9	1521534475691jzvzf	刘德华	2018/4/20 18:41	4
10	1521534475691jzvzf	赵雷	2018/4/20 20:13	4
11	1521534475691jzvzf	赵雷	2018/4/20 20:15	4
12	1521534475691jzvzf	赵雷	2018/4/20 20:16	4
13	1516925494173aQO5n	刘德华	2018/4/21 16:56	5
14	15232573243575bUSQ	周杰伦	2018/4/21 18:30	6
15	15242195457521H4J	林俊杰	2018/4/23 11:24	7
16	1523845728145zOfL	张学友	2018/4/23 11:44	8
17	1523413580330V4Wpf	许嵩	2018/4/23 11:55	1
18	1523845728145zOfL	许嵩	2018/4/23 11:56	8
19	1516925494173aQO5n	赵雷	2018/4/23 18:51	5
20	15234269125215oH4a	赵雷	2018/4/23 19:20	9
21	1521534475691jzvzf	赵雷	2018/4/23 19:20	4
22	15242195457521H4J	王力宏	2018/4/23 21:44	7
23	15234269125215oH4a	林俊杰	2018/4/24 10:20	9
24	1524100798053ik6re	梁静茹	2018/4/24 16:22	10
25	1524041544624GyRG	刘德华	2018/4/24 16:45	3

这个列表中，最后一列表示 UserTitle 映射的 UserID，然后我们再次将这份数据去掉重复的部分，前面我们对用户的 ID 进行了映射，同样因为歌手名也是字符

串类型的，为了方便处理，我们同样要像上述那样来对歌手名分配相应的 ID，代码如下：

```
singer4 = singer_recom.drop_duplicates(subset='SingerTitle', keep =
'first')

singer4['SingerID'] = range(0, singer4.shape[0])

columns = ["SingerTitle", "SingerID"]

singer_singer = singer4[columns]
```

处理之后，我们可以得到映射 ID 之后的歌手 ID：

singer_singer		
	SingerTitle	SingerID
0	邓丽君	0
1	赵雷	1
2	一朵	2
3	小虎队	3
4	刘德华	4
5	周杰伦	5
6	五月天	6
7	林俊杰	7
16	张学友	8
17	许嵩	9
22	王力宏	10
24	梁静茹	11
26	金志文	12
29	毛不易	13
30	谭维维	14
32	王菲	15
36	张杰	16
38	水木年华	17
39	张宇	18
40	腾格尔	19
47	杨丞琳	20
48	光良	21
49	薛之谦	22
62	蔡琴	23
68	戴荃	24
76	李健	25

当然了，这张图只是部分歌手的 ID 映射，同样的道理，我们要将歌手 ID 这一列加入到原来的数据列表中，于是我们可以进行如下的操作：

```
singer = singer_singer['SingerTitle'].values

singer_id = singer_singer['SingerID'].values

series_custom = Series(singer_id,index=singer)

trans_singer = []

for i in singer_recom["SingerTitle"]:

    trans_singer.append(series_custom[i])

singer_recom["SingerID"] = trans_singer
```

操作完成之后我们添加了”SingerID”这一列的数据，如下图所示：

singer_recom					
	UserTitle	SingerTitle	Date	UserID	SingerID
0	1523260590511gsTtu	邓丽君	2018/4/20 10:13	0	0
1	1523413580330V4Wpf	赵雷	2018/4/20 14:12	1	1
2	1522750485469mzXqM	一朵	2018/4/20 14:28	2	2
3	1524041544624GyRG	小虎队	2018/4/20 14:40	3	3
4	1521534475691jzvzf	刘德华	2018/4/20 16:15	4	4
5	1521534475691jzvzf	周杰伦	2018/4/20 16:30	4	5
6	1521534475691jzvzf	五月天	2018/4/20 16:30	4	6
7	1521534475691jzvzf	林俊杰	2018/4/20 18:38	4	7
8	1521534475691jzvzf	林俊杰	2018/4/20 18:38	4	7
9	1521534475691jzvzf	刘德华	2018/4/20 18:41	4	4
10	1521534475691jzvzf	赵雷	2018/4/20 20:13	4	1
11	1521534475691jzvzf	赵雷	2018/4/20 20:15	4	1
12	1521534475691jzvzf	赵雷	2018/4/20 20:16	4	1
13	1516925494173aQO5n	刘德华	2018/4/21 16:56	5	4
14	15232573243575bUSQ	周杰伦	2018/4/21 18:30	6	5
15	15242195457521H4J	林俊杰	2018/4/23 11:24	7	7
16	1523845728145zOfL	张学友	2018/4/23 11:44	8	8
17	1523413580330V4Wpf	许嵩	2018/4/23 11:55	1	9
18	1523845728145zOfL	许嵩	2018/4/23 11:56	8	9
19	1516925494173aQO5n	赵雷	2018/4/23 18:51	5	1
20	15234269125215oH4a	赵雷	2018/4/23 19:20	9	1
21	1521534475691jzvzf	赵雷	2018/4/23 19:20	4	1
22	15242195457521H4J	王力宏	2018/4/23 21:44	7	10
23	15234269125215oH4a	林俊杰	2018/4/24 10:20	9	7
24	1524100798053ik6re	梁静茹	2018/4/24 16:22	10	11
25	1524041544624GyRG	刘德华	2018/4/24 16:45	3	4

当然这只是一部分的数据，我们可以看到最后一列添加了歌手的 ID 值，接下来我们根据用户和歌手的 ID 对这些数据进行排序：

```
singer_recom = singer_recom.sort_values(["UserID", "SingerID"], inplace = False)
```

在排序之后，我们可以得到一个完整的歌手和用户的数据列表：

singer_recom					
	UserTitle	SingerTitle	Date	UserID	SingerID
0	1523260590511gsTtu	邓丽君	2018/4/20 10:13	0	0
1635	1523260590511gsTtu	邓丽君	2018/4/20 10:13	0	0
635	1523260590511gsTtu	周杰伦	2018-04-20 15:09:16	0	5
86	1523260590511gsTtu	林俊杰	2018/5/2 16:22	0	7
87	1523260590511gsTtu	林俊杰	2018/5/2 16:22	0	7
88	1523260590511gsTtu	林俊杰	2018/5/2 16:23	0	7
89	1523260590511gsTtu	林俊杰	2018/5/2 16:23	0	7
90	1523260590511gsTtu	林俊杰	2018/5/2 16:24	0	7
91	1523260590511gsTtu	林俊杰	2018/5/2 16:24	0	7

在得到这个完整的数据列表之后，我们并不是所有的数据都需要的，所以接下来我们保留有用的特征数据，将 UserID 和 SingerID 进行聚合操作，这里用到的是聚合函数 pandas 中的 groupby：

```
target = singer_recom.groupby(['UserID', 'SingerID']).size().reset_index()
```

在聚合了 UserID 和 SingerID 之后，我们可以用 reset\_index() 函数将 groupby 之后的数据使用 size() 方法统计使用评率之后，转换成 DataFrame 对象，处理



之后我们可以得到如下结果：

target			
	UserID	SingerID	0
0	0	0	2
1	0	5	1
2	0	7	14
3	0	8	1
4	0	18	2
5	0	37	2
6	0	41	1
7	1	1	4
8	1	5	3
9	1	7	1

然后我们可以将每个歌手的频次标注出来，然后听的歌手的频次就是我们对这个歌手的打分，打分系统是一个复杂的系统，我们暂且很简单地认为用户听哪个歌手的歌多，就对这个歌手的评分就高：

```
target.columns = ['UserID', 'SingerID', 'Rating']
```

最后我们可以得到含有打分项的列表：

target			
	UserID	SingerID	Rating
0	0	0	2
1	0	5	1
2	0	7	14
3	0	8	1
4	0	18	2
5	0	37	2
6	0	41	1
7	1	1	4
8	1	5	3
9	1	7	1

为了方便我们对照歌手名，我们将歌手名和用户名 merge 到这张表中来：

```
df_user = pd.merge(target, singer_singer, on = 'SingerID', how = 'left',  
suffixes = ('_', ''))  
  
df_user = pd.merge(df_user, singer_user, on = 'UserID', how = 'left',  
suffixes = ('_', ''))
```

在 merge 完成之后，我们的数据预处理就完成了，就可以得到我们需要的数据格式了，然后我们把预处理完成之后的.csv 数据储存起来：

```
df_user = pd.merge(df_user, singer_user, on = 'UserID', how = 'left', suffixes = ('_', ''))
```

df\_user

	UserID	SingerID	Rating	SingerTitle	UserTitle
0	0	0	2	邓丽君	1523260590511gsTtu
1	0	5	1	周杰伦	1523260590511gsTtu
2	0	7	14	林俊杰	1523260590511gsTtu
3	0	8	1	张学友	1523260590511gsTtu
4	0	18	2	张宇	1523260590511gsTtu
5	0	37	2	陈奕迅	1523260590511gsTtu
6	0	41	1	李宗盛	1523260590511gsTtu
7	1	1	4	赵雷	1523413580330V4Wpf
8	1	5	3	周杰伦	1523413580330V4Wpf

```
df_user.to_csv("dataset/singer_preprocess_result.csv", encoding = "utf-8")
```

上面就是我们拿到数据之后到预处理完成之后的过程，接下来我们就是进行推荐算法的实现，这里我们用到的是 pyspark，首先我们导入 pyspark 库，然后读取我们已经经过预处理之后的数据：

```
from pyspark.sql import SparkSession  
  
spark =  
SparkSession.builder.appName("singer_recommendation").master("local").g  
etOrCreate()
```



```
df = spark.read.format('com.databricks.spark.csv').options(header = 'true',  
inferschema = 'true').load("dataset/singer_preprocess_result.csv")
```

在导入了数据之后,我们再来进行特征的提取,在 spark 中直接可以使用 select() 方法:

```
df_singer_recommend = df.select("UserID", "SingerID", "Rating")
```

在进行完特征提取之后,我们这里就要用到 spark 的 rdd 了,什么是 RDD? RDD (Resilient Distributed Dataset) 叫做弹性分布式数据集,是 Spark 中最基本的数据抽象,它代表一个不可变、可分区、里面的元素可并行计算的集合。RDD 具有数据流模型的特点:自动容错、位置感知性调度和可伸缩性。RDD 允许用户在执行多个查询时显式地将工作集缓存在内存中,后续的查询能够重用工作集,这极大地提升了查询速度,创建 rdd 的代码如下:

```
singer_rdd = df_singer_recommend.rdd  
  
trainingRDD = singer_rdd.cache()
```

这里运用到了 rdd 的缓存 cache 方法,这样可以将 rdd 缓存到磁盘或者内存中,以便于后续的复用。准备好了这些之后,我们就可以开始训练模型了,首先我们要导入 spark 的库:

```
from pyspark.mllib.recommendation import ALS, MatrixFactorizationModel,  
Rating
```

这里用到的是 pyspark 库中的 ALS 算法,然后我们就可以去训练一个推荐系统的模型:

```
rank = 10  
  
numIterations = 10  
  
model = ALS.train(trainingRDD, rank, numIterations)
```

这里我们来看看 ALS 中的 train 方法,这里的 rank 表示隐藏因子的个数, numIterations 表示迭代的次数,然后 trainingRDD 表示我们训练的数据源,这是我们 pyspark 中的 API,很方便使用的,然后训练完了之后我们就可以得到我们的模型 model,这里我们再来简单地看看 ALS 算法是怎么回事,ALS 算法

其实是交替最小二乘法，比如用户听歌，如果每个歌手都是一个维度，但是用户只会去听部分自己喜欢的歌手的歌，所以在这整个矩阵中，只会有很小的一部分会有值，大部分的地方都是没有值的，我们就认为这个矩阵就是一个稀疏矩阵，然后我们要想办法将这个矩阵的数据填满，让她形成一个稠密的矩阵，操作起来就是说通过降维的方法来补全矩阵，对矩阵没有出现的值进行估值，场用的方法就是 **SVD**（奇异值分解），这个方法在矩阵分解之前需要先把评分矩阵 **R** 缺失值补全，补全之后稀疏矩阵 **R** 矩阵表示成稠密矩阵 **R'**，然后我们可以将 **R'** 矩阵进行分解成：

$$R'=UTSV$$

在这个公式中，假如 **R'** 是一个  $m \times n$  的矩阵，那么可以分解成，**U** 矩阵的转秩  $m \times k$ ，**S** 矩阵  $k \times s$ ，和 **V** 矩阵  $s \times n$  矩阵相乘，这个公式中，选取 **U** 中的 **K** 列和 **V** 中的 **s** 行作为隐特征数，从而达到降维的目的。**ALS** 推荐算法大概的思想就是这个样子，我后续会写一篇关于 **ALS** 的算法的详细介绍，大家一起学习，这里就点到为止。接下来我们定义一个方法来返回 **top5** 的商品 ID：

```
def top5_productID(userID):  
  
    recommendedResult = model.recommendProducts(userID, 5)  
  
    product_id_list = []  
  
    for i in range(5):  
  
        product_id_list.append(recommendedResult[i].product)  
  
    return product_id_list
```

这里就是利用 **spark** 的 **API** 中的 **recommendProducts** 方法来返回排名最高的 5 个商品 ID。有了这个 **list**，我们再来构建一个 **dataframe**：

```
def construct_dataframeData(userList):  
  
    data = []  
  
    for user in userList:  
  
        res = top5_productID(user)
```

```
res.insert(0, user)

data.append(res)

return data
```

通过这个方法，我们就可以将每一个用户推荐的 5 个商品，也就是将每一个用户所推荐的 5 个歌手插入到 dataframe 中：

```
data = construct_dataframeData(unique_userid_list)
```

接着我们可以得到一个结构体 dataframe:

```
data
[[0, 7, 33, 22, 9, 50],
 [1, 71, 1, 82, 12, 5],
 [2, 9, 1, 4, 5, 7],
 [3, 1, 4, 33, 52, 55],
 [4, 5, 7, 16, 6, 4],
 [5, 33, 9, 6, 23, 52],
 [6, 7, 33, 5, 9, 62],
 [7, 9, 6, 33, 53, 28],
 [8, 26, 9, 6, 4, 7],
 [9, 9, 6, 5, 22, 28],
 [10, 33, 4, 15, 1, 52],
 [11, 9, 1, 5, 30, 6],
 [12, 9, 6, 33, 23, 5],
 [13, 1, 5, 4, 23, 7],
 [14, 1, 30, 33, 22, 12],
 [15, 22, 15, 26, 9, 33],
 [16, 9, 22, 6, 30, 5],
 [17, 9, 33, 6, 22, 23],
 [18, 9, 33, 6, 7, 27],
 [19, 7, 22, 5, 26, 30],
 [20, 9, 33, 22, 30, 7],
 [21, 22, 30, 5, 26, 23],
 [22, 9, 6, 5, 28, 13],
 [23, 1, 9, 5, 55, 30],
 [24, 5, 6, 82, 53, 13],
 [25, 5, 23, 4, 9, 1],
 [26, 9, 7, 5, 6, 26],
 [27, 6, 9, 22, 5, 53],
 [28, 1, 7, 4, 5, 33],
 [29, 22, 23, 6, 30, 33],
 [30, 9, 7, 22, 33, 15],
 [31, 5, 9, 4, 16, 6],
 [32, 6, 22, 5, 9, 7],
 [33, 26, 6, 9, 4, 8],
 [34, 1, 5, 4, 23, 6],
 [35, 22, 6, 5, 7, 33],
 [36, 6, 9, 53, 52, 28],
 [37, 6, 9, 22, 53, 33],
 [38, 23, 6, 33, 1, 53],
 [39, 52, 33, 9, 1, 4],
 [40, 1, 4, 33, 55, 16],
 [41, 33, 9, 6, 23, 7],
 [42, 7, 5, 9, 6, 33],
 [43, 82, 6, 71, 53, 7],
 [44, 1, 5, 23, 4, 55],
 [45, 23, 9, 5, 1, 4],
 [46, 6, 1, 53, 5, 23],
 [47, 22, 6, 23, 30, 5],
 [48, 22, 6, 30, 1, 53],
 [49, 23, 22, 5, 6, 33],
 [50, 33, 9, 7, 6, 119],
 [51, 23, 5, 9, 22, 31]]
```

这个就是我们推荐结果矩阵中的一部分结果，第一个数字表示用户的 UserID，后面的 5 个数字表示推荐给用户的 SingerID，其实这个矩阵就相当于

是我们的推荐结果，我们接下来需要做的就是将这个矩阵解析成我们可以看得懂的文本矩阵输出。所以我们再写一个方法，将我们数字的矩阵映射成文本的矩阵结果输出：

```
def parse_data(dataList):  
  
    res = []  
  
    for item in dataList:  
  
        res_item = []  
  
        UserTitle = df.filter(df['UserID'] ==  
item[0]).select("UserTitle").collect()[0].UserTitle  
  
        res_item.append(UserTitle)  
  
        for i in range(1, 6):  
  
            SingerTitle = df.filter(df['SingerID'] ==  
item[i]).select("SingerTitle").collect()[0].SingerTitle  
  
            res_item.append(SingerTitle)  
  
        res.append(res_item)  
  
    return res
```

通过这个方法，我们将数字矩阵可以转化成文本矩阵，我们来看看最终推荐的部分结果吧：

parseData

```
[['1523260590511gsTtu', '林俊杰', '张碧晨', '薛之谦', '许嵩', '黄家驹'],
['1523413580330V4Wpf', '那英', '赵雷', '鹿晗', '金志文', '周杰伦'],
['1522750485469mzXqM', '许嵩', '赵雷', '刘德华', '周杰伦', '林俊杰'],
['1524041544624GyRG', '赵雷', '刘德华', '张碧晨', '李玉刚', '娃娃'],
['1521534475691jzvzf', '周杰伦', '林俊杰', '张杰', '五月天', '刘德华'],
['1516925494173aQ05n', '张碧晨', '许嵩', '五月天', '蔡琴', '李玉刚'],
['15232573243575bUSQ', '林俊杰', '张碧晨', '周杰伦', '许嵩', '阿杜'],
['15242195457521H4J', '许嵩', '五月天', '张碧晨', '祁隆', '汪峰'],
['1523845728145z0fL', '任贤齐', '许嵩', '五月天', '刘德华', '林俊杰'],
['15234269125215oH4a', '许嵩', '五月天', '周杰伦', '薛之谦', '汪峰'],
['1524100798053ik6re', '张碧晨', '刘德华', '王菲', '赵雷', '李玉刚'],
['1516777944799JwgsF', '许嵩', '赵雷', '周杰伦', '游鸿明', '五月天'],
['1523585081511aqccN', '许嵩', '五月天', '张碧晨', '蔡琴', '周杰伦'],
['15247029781055xsa', '赵雷', '周杰伦', '刘德华', '蔡琴', '林俊杰'],
['1506483949540Ihb6v', '赵雷', '游鸿明', '张碧晨', '薛之谦', '金志文'],
['1501837243237IJhHm', '薛之谦', '王菲', '任贤齐', '许嵩', '张碧晨'],
['1510119557992AsBC7', '许嵩', '薛之谦', '五月天', '游鸿明', '周杰伦'],
['1524626841265fvqp', '许嵩', '张碧晨', '五月天', '薛之谦', '蔡琴'],
['1523417280227IicMe', '许嵩', '张碧晨', '五月天', '林俊杰', '伍佰'],
['1523933385072hJ1c', '林俊杰', '薛之谦', '周杰伦', '任贤齐', '游鸿明'],
['15252583675624PZo', '许嵩', '张碧晨', '薛之谦', '游鸿明', '林俊杰'],
['1512438589203IaCrr', '薛之谦', '游鸿明', '周杰伦', '任贤齐', '蔡琴'],
['1500255789400D8zPl', '许嵩', '五月天', '周杰伦', '汪峰', '毛不易'],
['1525317740134mNOV', '赵雷', '许嵩', '周杰伦', '娃娃', '游鸿明'],
['1520301661697ignar', '周杰伦', '五月天', '鹿晗', '祁隆', '毛不易'],
['1499219097530LzUTe', '周杰伦', '蔡琴', '刘德华', '许嵩', '赵雷'],
['1494222003785hUyQY', '许嵩', '林俊杰', '周杰伦', '五月天', '任贤齐'],
['1525655582638qbJm', '五月天', '许嵩', '薛之谦', '周杰伦', '祁隆'],
['15253382002576sqZ', '赵雷', '林俊杰', '刘德华', '周杰伦', '张碧晨'],
['1525259087523CwNp', '薛之谦', '蔡琴', '五月天', '游鸿明', '张碧晨'],
['1524877112171J7ut', '许嵩', '林俊杰', '薛之谦', '张碧晨', '王菲'],
['15014664651329qx5v', '周杰伦', '许嵩', '刘德华', '张杰', '五月天'],
['1525914719219Y0SJ', '五月天', '薛之谦', '周杰伦', '许嵩', '林俊杰'],
['15014663963532hf8j', '任贤齐', '五月天', '许嵩', '刘德华', '张学友'],
['15252623804099Qy4', '赵雷', '周杰伦', '刘德华', '蔡琴', '五月天'],
['1494032773553dLx0q', '薛之谦', '五月天', '周杰伦', '林俊杰', '张碧晨'],
['1502157924095XdwM5', '五月天', '许嵩', '祁隆', '李玉刚', '汪峰'],
['1525773202537zQVh', '五月天', '许嵩', '薛之谦', '祁隆', '张碧晨'],
['1522290399591Hz46G', '蔡琴', '五月天', '张碧晨', '赵雷', '祁隆'],
['1493868900256test1', '李玉刚', '张碧晨', '许嵩', '赵雷', '刘德华'],
['1526729828297BQVF', '赵雷', '刘德华', '张碧晨', '娃娃', '张杰'],
['1495009142622SR7h8', '张碧晨', '许嵩', '五月天', '蔡琴', '林俊杰'],
['1526980145096kKpy7', '林俊杰', '周杰伦', '许嵩', '五月天', '张碧晨'],
['1494228599129DYKOi', '鹿晗', '五月天', '那英', '祁隆', '林俊杰'],
['1526622936275K5TH', '赵雷', '周杰伦', '蔡琴', '刘德华', '娃娃'],
['1526633579156z14u', '蔡琴', '许嵩', '周杰伦', '赵雷', '刘德华'],
['15279071735007kON', '五月天', '赵雷', '祁隆', '周杰伦', '蔡琴'],
['1494336342716mybjD', '薛之谦', '五月天', '蔡琴', '游鸿明', '周杰伦'],
['1524912104719HE2G', '薛之谦', '五月天', '游鸿明', '赵雷', '祁隆']]
```

这样就可以看到每一个用户所对用的最喜欢的 5 个歌手，所以我们就得到了推荐最终的结果，我们可以将这些结果导出成.csv：

```
df_res = pd.DataFrame(parseData, columns = ['UserTitle', 'SingerName1',
'SingerName2', 'SingerName3', 'SingerName4', 'SingerName5'])

df_res.to_csv("singer_recommend_result.csv", index=False,
encoding="utf-8")
```

有了这个推荐列表，我们可以将这个列表上线到我们的后台服务中去。到这里我们就可以说利用 spark 完成了一个完整的歌手推荐系统。

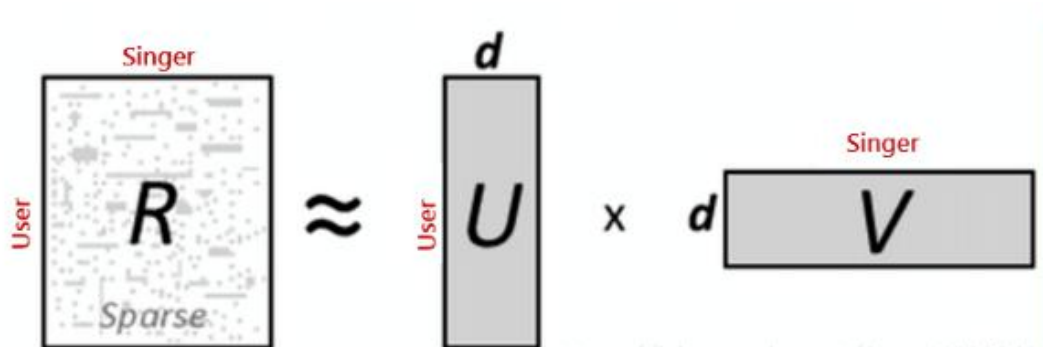
## 2.ALS 调参

在模型训练的时候，我们用到的是 ALS 算法，这篇博文我们就一起来学习一下 ALS 算法的原理吧。ALS 算法全称是 Alternating Least Squares，从协同过滤的分类来说，这里的 ALS 算法是同时基于用户和物品的协同，所以说 ALS 算法也是混合的协同过滤。其实用户和物品的关系可以由用户、物品和评分三项联系在一起，可以用<user, item, rating>来表示，rating 表示用户对商品的喜爱程度，也就是用户对商品的评分，在我的歌手推荐系统中，我们将用户听某首歌的频次作为这首歌的评分。当我们的歌手和用户的数量足够大的时候，我们就会得到一个非常大的矩阵，但是这个矩阵只有小部分的位置有数据，因此这个评分矩阵是一个稀疏矩阵，因为这个矩阵巨大，用传统的分解方法已经很难将这个矩阵进行分解了。所以我们假设用户和歌手之间存在着若干有联系的隐形特征，也就是维度，比如说用户的年纪，性别，歌手的年纪，用户的学历等等，这些都是一些隐形的特征，所以我们需要将原始的 R 矩阵投射在这些隐形的矩阵上面，用数学表达式可以表达成：

$$R_{m \times n} \approx X_{m \times k} \cdot Y_{n \times k}^T$$

这里的约等于符号表示这是一个近似的空间变换，也就是说原始的矩阵可以表示成  $m \times k$  和  $k \times n$  的矩阵相乘，这里的  $k$  就表示我们的隐藏因子数，也称作 latent factor，通过矩阵的拆分，我们可以做到数据降维的目的，矩阵拆分可以由下图表示：





我们的目的是让矩阵  $X$  和  $Y$  相乘尽可能地相似  $R$  矩阵，所以我们目的就是最小化我们矩阵的平方损失函数：

$$\min_{x^*, y^*} \sum_{u, i} (r_{ui} - x_u^T y_i)^2$$

整体来说，就是让我们整体的真实值与我们相乘的预测值尽可能小。我们考虑到矩阵稳定性的问题，我们可以将上述公式正则化：

$$\min_{x^*, y^*} L(X, Y) = \min_{x^*, y^*} \sum_{u, i} (r_{ui} - x_u^T y_i)^2 + \lambda(|x_u|^2 + |y_i|^2)$$

优化这个公式，最小化我们的损失函数，可以得到我们的  $X$  和  $Y$  矩阵，将这两个矩阵相乘，可以得到我们预测的矩阵，这个矩阵不仅可以预测我们用户对歌手的打分，还可以比较不同用户或者歌手之间的相似度。**ALS** 算法是一个离线算法，所以我们新添加了用户或者歌手，不能实时进行推荐，这是一个冷启动的问题，但是 **ALS** 算法在推荐系统中还是占有很重要的一席之地。

还有就是我们如何优化上述的公式，如何去最小化我们的损失函数。这里大家想到的就是将上述公式进行求导，但是这个公式直接求导是比较麻烦的，因为有  $X$  和  $Y$  两个变量，所以我们可以先优化一个维度，固定其他的维度，所以我们可以对  $x_u$  求偏导，然后我们可以得到如下的结果：

$$\frac{\partial L}{\partial x_u} = \frac{\partial}{\partial x_u} \sum_{u, i} (r_{ui} - x_u^T y_i)^2 + \lambda(|x_u|^2 + |y_i|^2) = 2 \sum_i (r_{ui} - x_u^T y_i) \frac{\partial}{\partial x_u} (-x_u^T y_i) + 2\lambda x_u =$$

求导完成之后，我们可以令这个导数为 0，然后可以得到下面的公式：

$$Y^T Y x_u + \lambda I x_u = Y^T r_u \Rightarrow x_u = (Y^T Y + \lambda I)^{-1} Y^T r_u$$

同样的道理，我们如果固定  $X$ ，然后对  $y_i$  求偏导，可以得到  $y_i$ ：



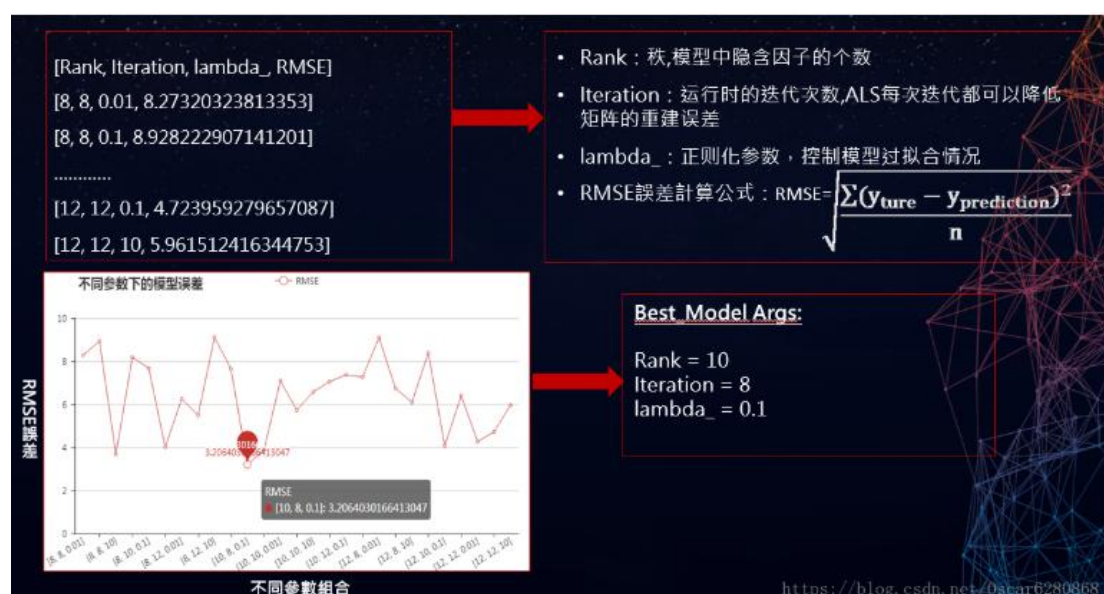
$$y_i = (X^T X + \lambda I)^{-1} X^T r_i$$

我们将  $x$  和  $y$  不断地进行迭代，一般来说，我们运用均方误差来评估是否收敛：

$$RMSE = \sqrt{\frac{\sum (R - XY^T)^2}{N}}$$

这个公式中的  $N$  表示<user, singer, rating>三元组的个数，当 RMSE 值迭代变化的值小于某个阈值的时候，我们就可以认为收敛了。

整个调参的过程我们可以由下面这个图表示：



整个调参过程如上图所示，我们得到最佳的参数，就可以进行模型训练了。