**

**HUNAN UNIVERSITY**

《数据仓库技术与应用》

实验报告九

|  |  |
| --- | --- |
| **报告名称：** | 基于TMDB数据集的电影数据分析 |
| **学生姓名：** | 杨超然 |
| **学生学号：** | 202106060220 |
| **专业班级：** | 电商2102班 |
| **学 院：** | 工商管理学院 |
| **指导老师：** | 周中定 |
| **日 期：** | 2023.4.17 |

目录

[一、实验过程 3](#_Toc2071910220)

[（一）实验环境配置 3](#_Toc1147731435)

[（二） 数据预处理 3](#_Toc1224110691)

[2.1数据集下载 3](#_Toc735045377)

[2.2csv文件处理 4](#_Toc1581713695)

[（三） 使用Spark将数据转换为DataFrame 4](#_Toc162005652)

[3.1完整代码 4](#_Toc1967212415)

[（四） 使用Spark进行数据分析 5](#_Toc280829693)

[4.1数据分析代码 5](#_Toc1883077792)

[完整数据分析代码如下（代码解释放在学习小记中）： 5](#_Toc1421944305)

[（五） 数据可视化 7](#_Toc1419910319)

[（六） 数据图表 10](#_Toc1594445969)

[6.1概览类图表 10](#_Toc1552453717)

[6.2关系类图表 14](#_Toc163310569)

[二、易错点总结 18](#_Toc276632317)

[（一）使用spark-submit命令时报错 18](#_Toc57256064)

[三、学习小记 18](#_Toc229993792)

[（一）实验心得 18](#_Toc35097544)

[（二）数据分析代码的相关解释 19](#_Toc1473902730)

[2.1.TMDB电影中的体裁分布 19](#_Toc659314965)

[2.2前100个常见关键词 20](#_Toc90998235)

[2.3TMDB中最常见的十种预算数 20](#_Toc398978981)

[2.4TMDB中常见电影时长 21](#_Toc1195787733)

[2.5生产电影最多的10大公司 21](#_Toc1452459905)

[2.6TMDB中的十大电影语言 21](#_Toc1047007886)

[2.7预算与评价的关系 22](#_Toc580536484)

[2.8发行时间与评价的关系 22](#_Toc1058478267)

[2.9流行度与评价的关系 22](#_Toc89701721)

[2.10公司发行电影平均分与数量的关系 22](#_Toc83304653)

[2.11电影预算和营收的关系 23](#_Toc2089448774)

[（三）数据转换代码的相关解释 23](#_Toc1712948874)

[3.1文件转化 23](#_Toc365953636)

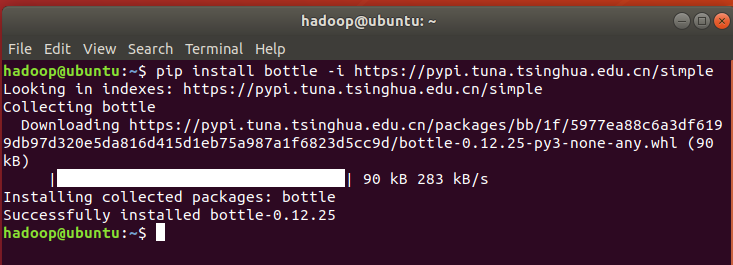
[3.2创建用于转为DataFrame的RDD 24](#_Toc189595244)

# 一、实验过程

## （一）实验环境配置

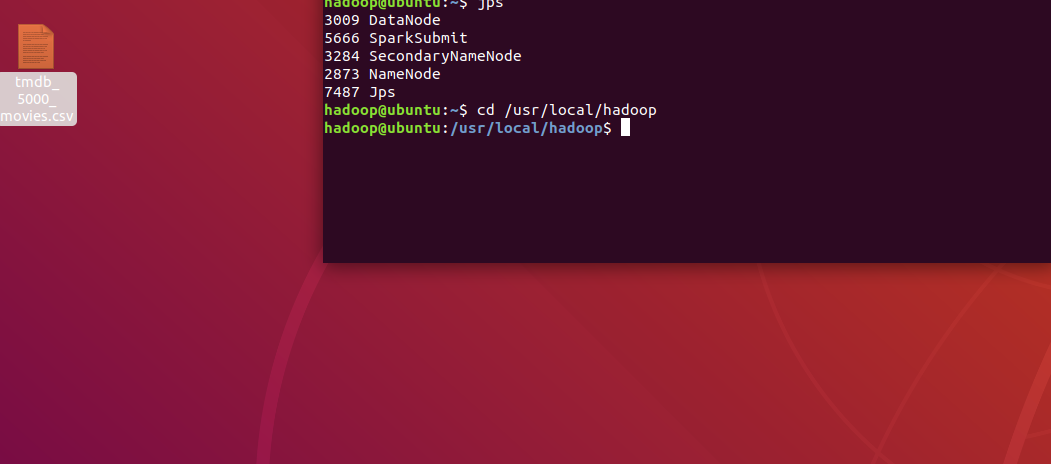
本次实验环境：Linux：Ubuntu 16.04；Hadoop3.1.3；Python：3.6；Spark：2.4.0；Jupyter Notebook Bottle相关安装操作见实验六。

配置web框架bottle的过程如下：



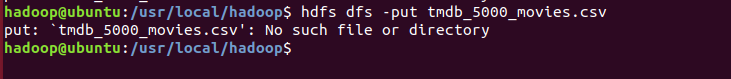
## 数据预处理

2.1数据集下载

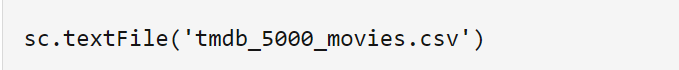
本次实验数据集来自Kaggle的tmdb-movie-metadata电影数据集，使用tmdb\_5000\_movies.csv进行实验：  


2.2csv文件处理

为了更方便将csv文件转换为RDD进行处理，需要首先去除csv文件的标题行，然后，将处理好的csv文件上传到HDFS上方便进一步处理：



此时文件在 HDFS 上的路径为 ：/user/hadoop/tmdb\_5000\_movies.csv，之后在程序中，使用下面语句即可读取该文件：



## 使用Spark将数据转换为DataFrame

3.1完整代码

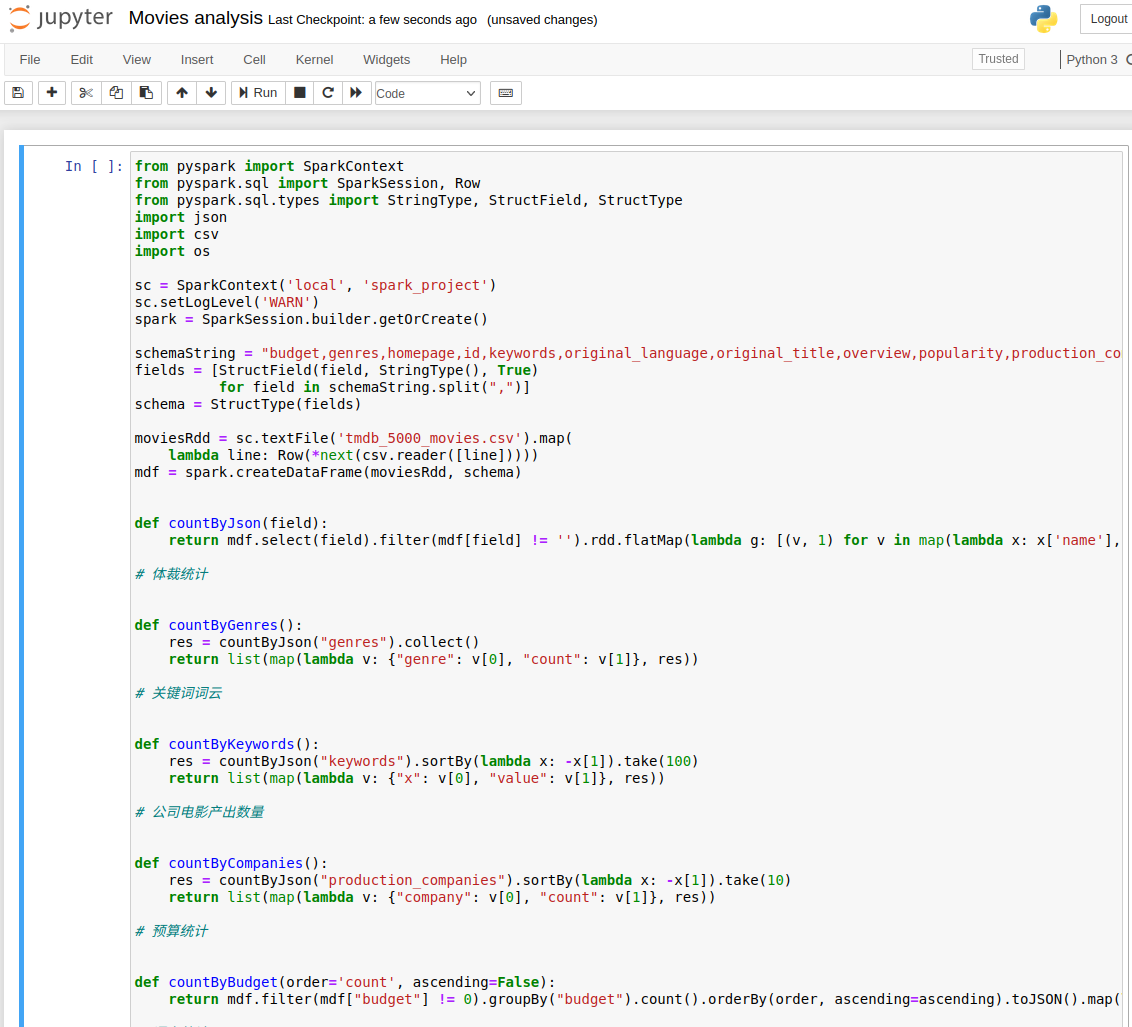
整体代码如下（代码解释放在学习小记中）：



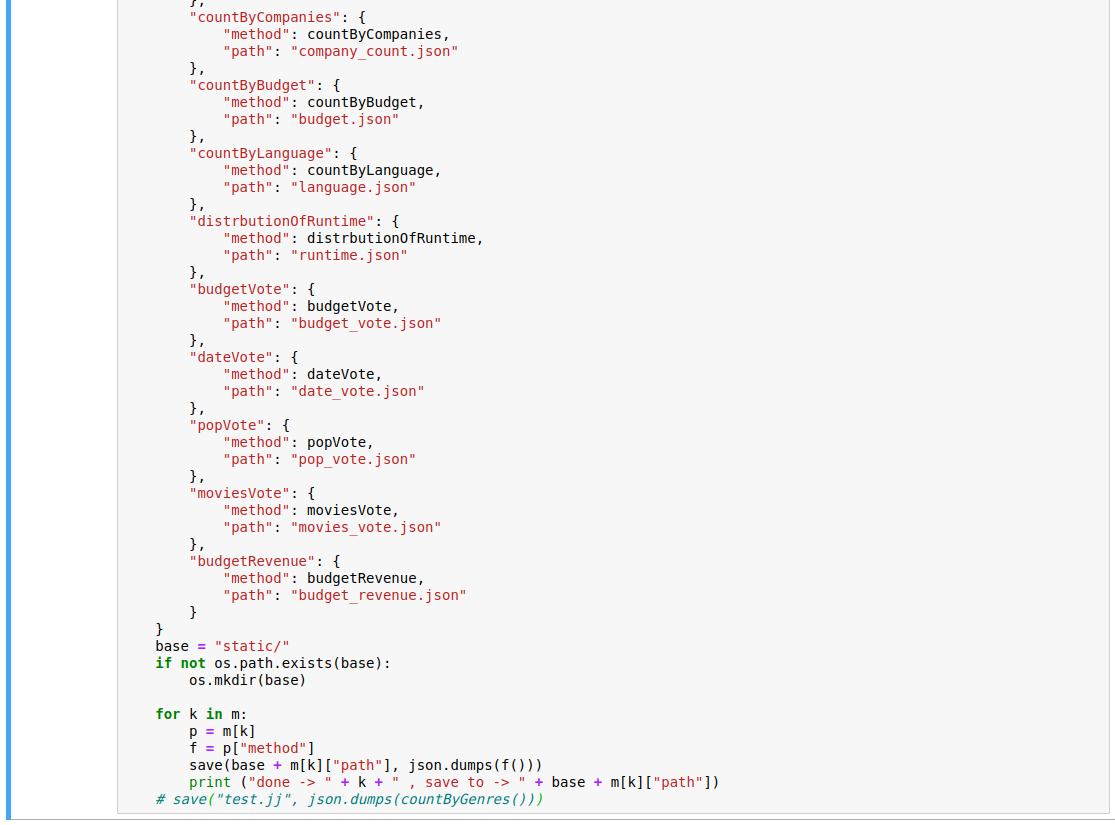
## 使用Spark进行数据分析

4.1数据分析代码

完整数据分析代码如下（代码解释放在学习小记中）：



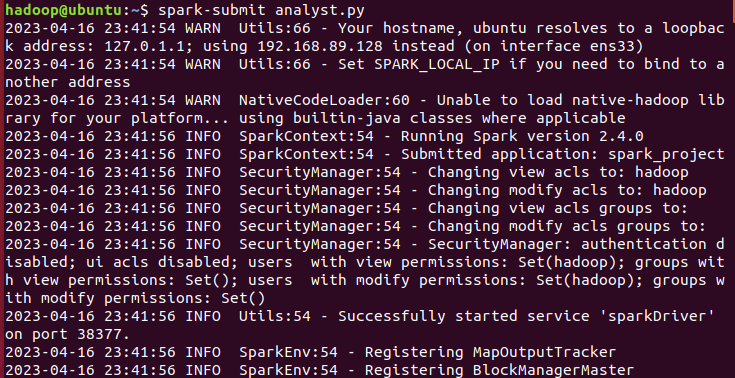


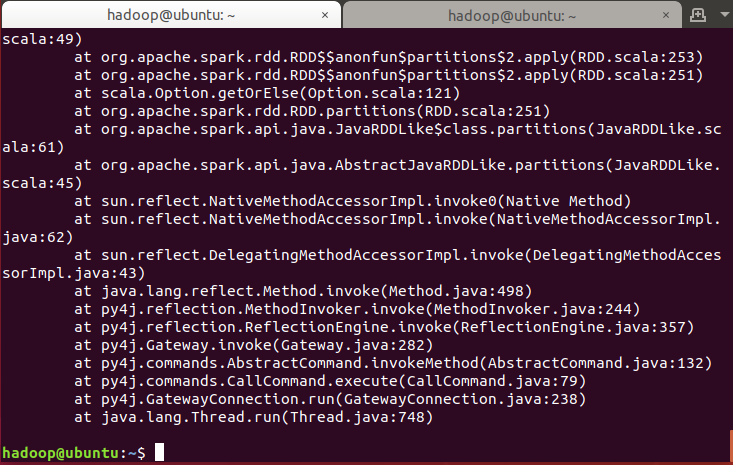


或在terminal中，采用如下方法进行数据分析：



运行部分结果如下：





## 数据可视化

首先使用 python Web 框架 bottle 访问可视化页面方便进行 json 数据的读取。使用下面代码web.py 可以实现一个简单的静态文件读取：

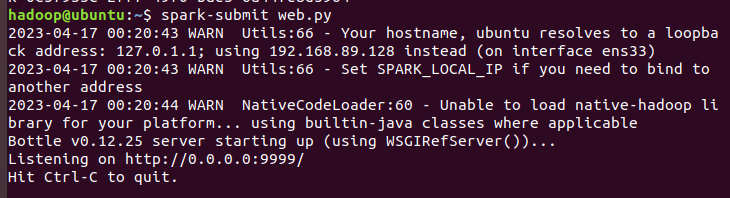


接下来实现主页文件 index.html：

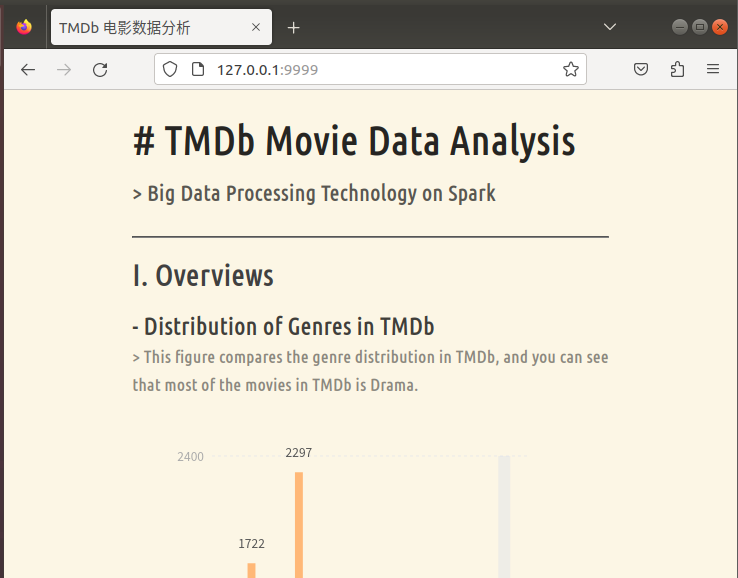


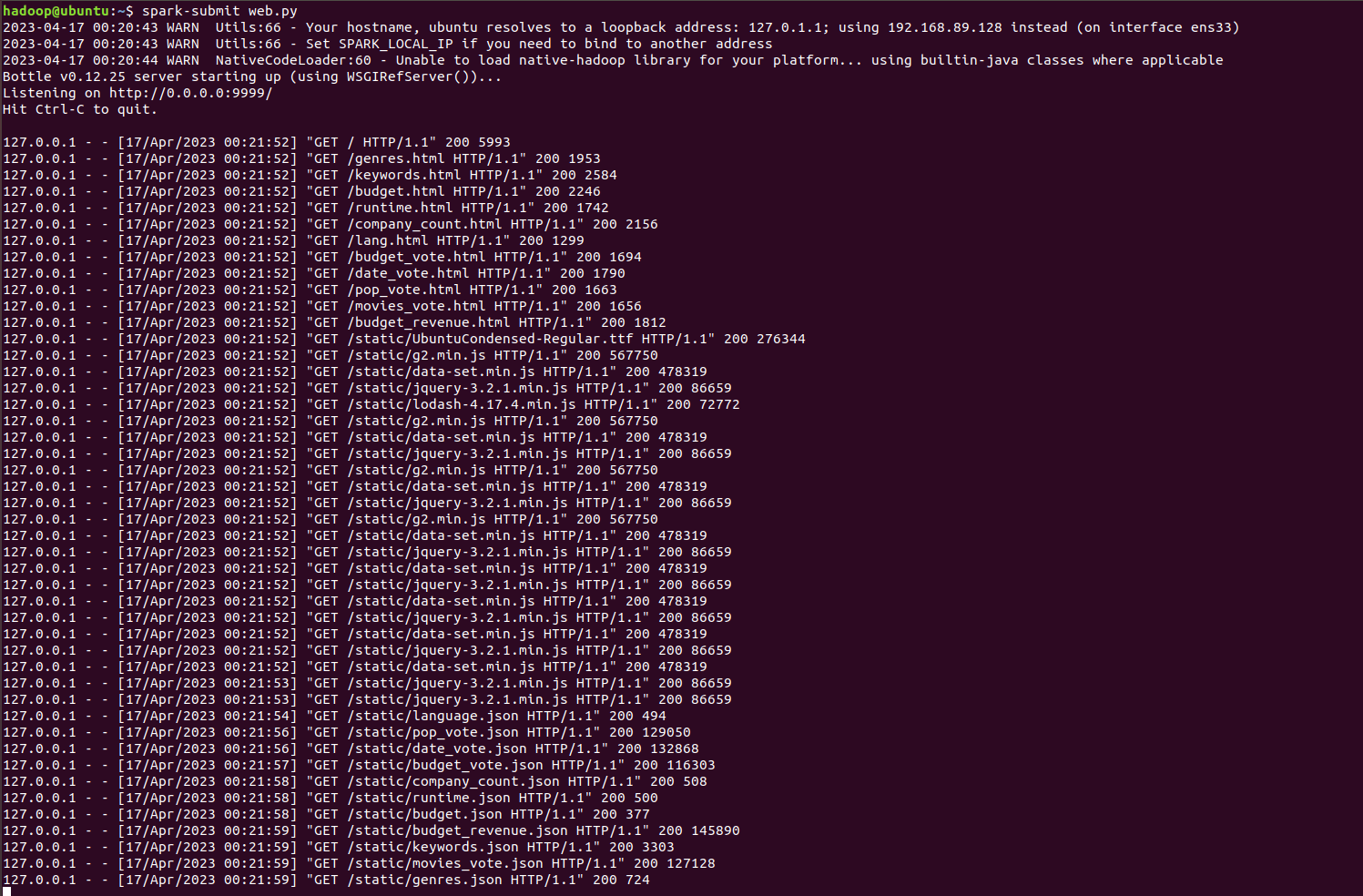
每个图表通过一个 iframe 引入到主页中。对于每一个图表，主页中都包含标题和图表所在的页面的 iframe。对于 TMDb 中的体裁分布分析结果，在 genres.html 中实现，下面对该文件进行实现：

使用该页面前，还需要将对应的 js 库( g2.js, data-set.js, jquery )放入到 static 文件夹下。代码完成后，在代码所在的根目录下，执行：



出现上图界面，即完成启动，打开浏览器访问 http://127.0.0.1:9999 即可看到可视化结果：



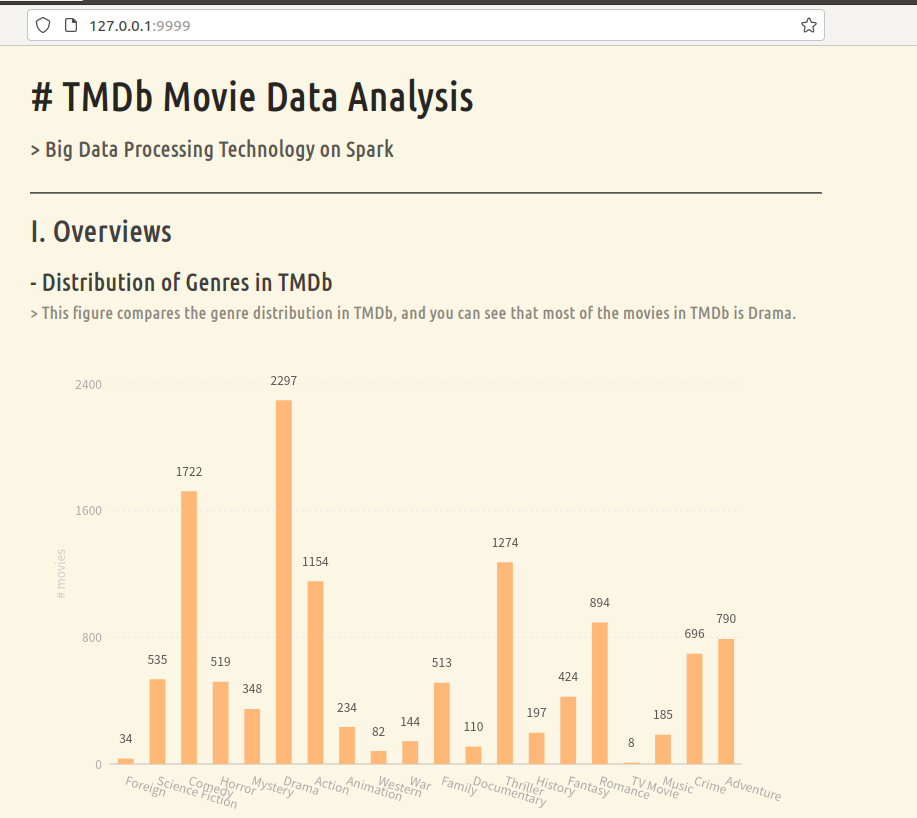


具体结果展示将会放在（六）中。

## 数据图表

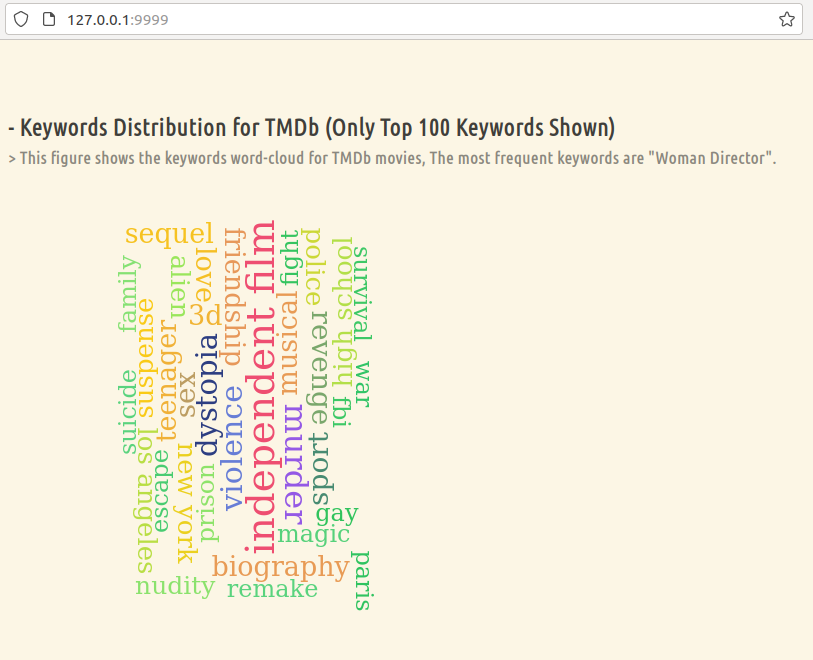
6.1概览类图表

6.1.1TMDB电影中的体裁分布



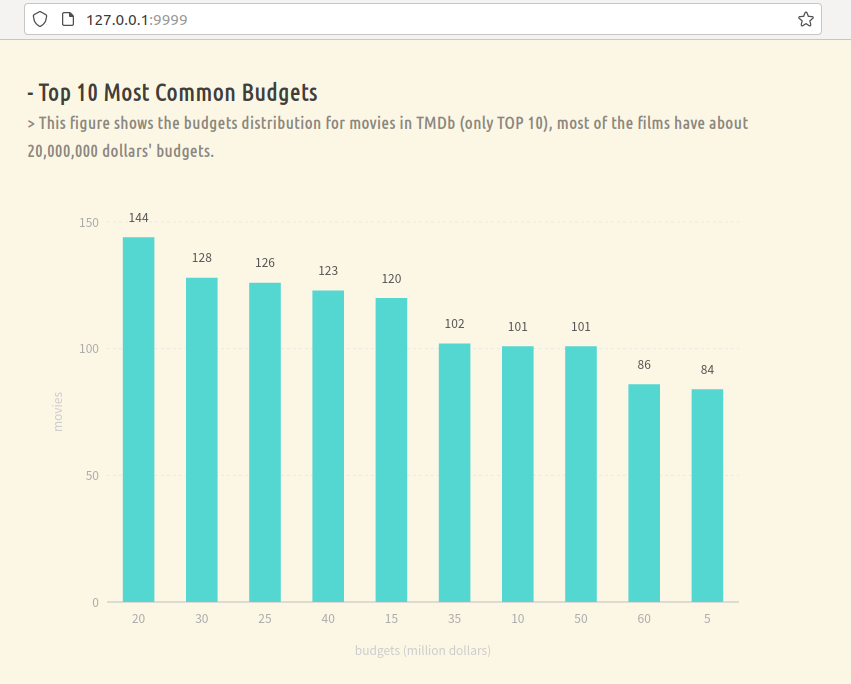
从图中可以看出，Drama 的电影在 TMDb 中占比较大，其次 Science Fiction、Action 和 Thriller 的数量也较多。

6.1.2前100个常见关键词



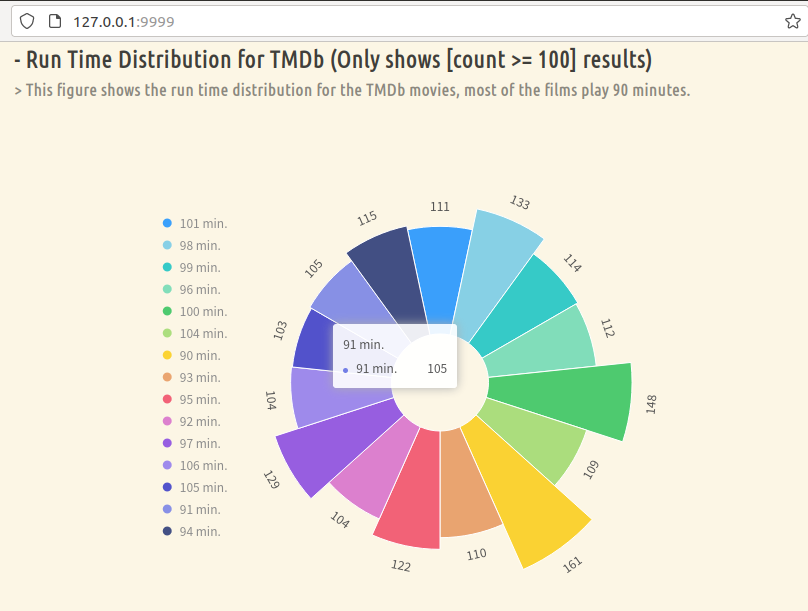
TMDb 中最常见的关键词是 Woman Director，其次还有 independent film 等。

6.1.3TMDB中常见的10种预算数



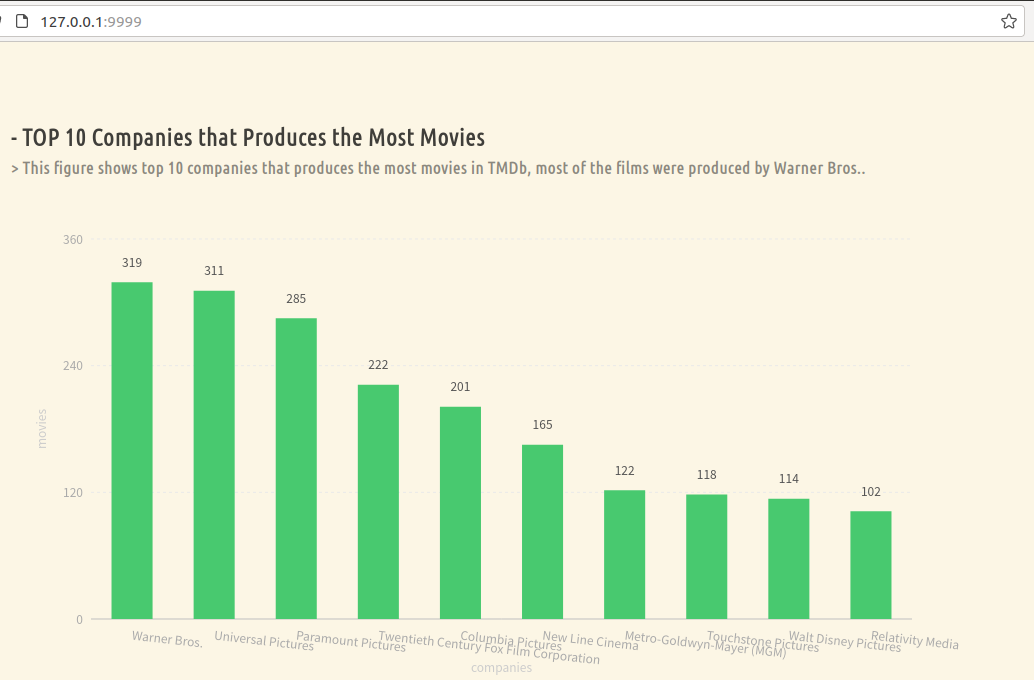
有 144 部电影的预算为 20,000,000，是最常见的预算值。

6.1.4TMDB 中最常见电影时长 (只展示电影数大于 100 的时长)



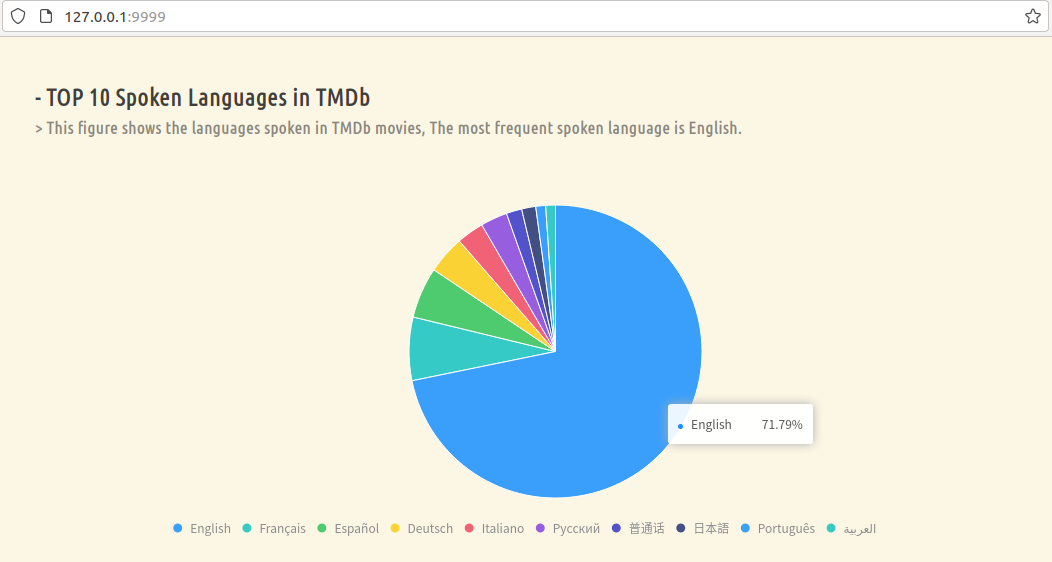
可见多数电影的时长是90分钟或100分钟。

6.1.5生产电影最多的十大公司



可见生产电影较多的公司是 Warner Bros.、Universal Pictures等

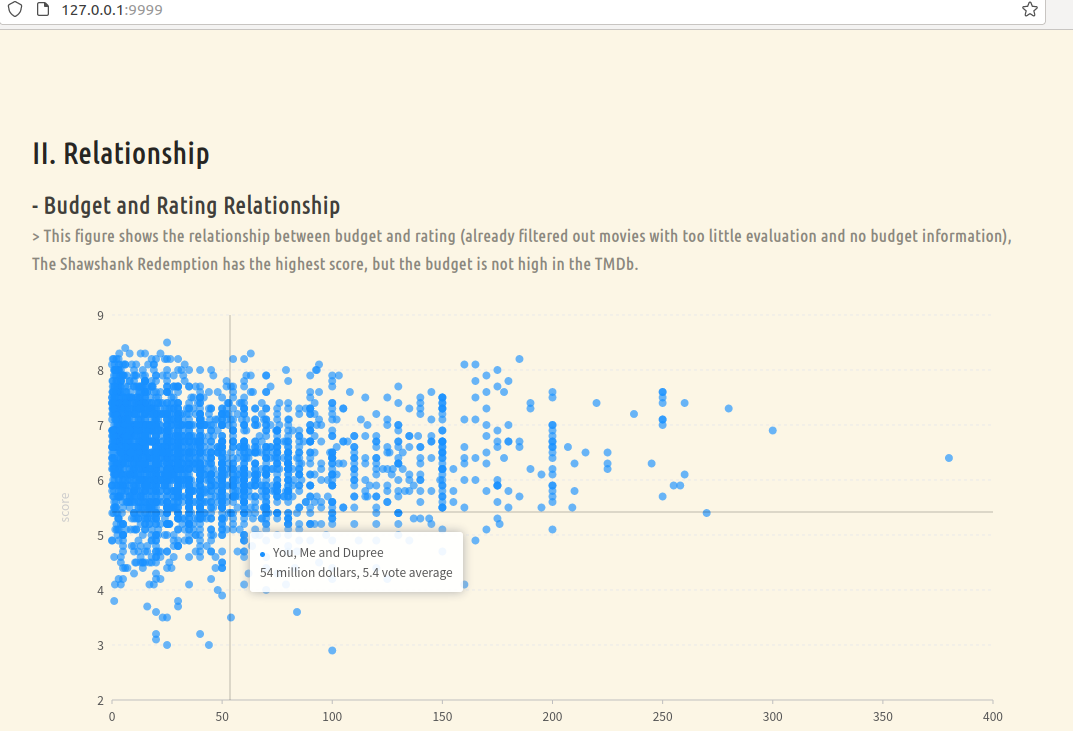
6.1.6TMDB中的十大电影语言



可见英语是占比最大的电影语言，即71.79%。

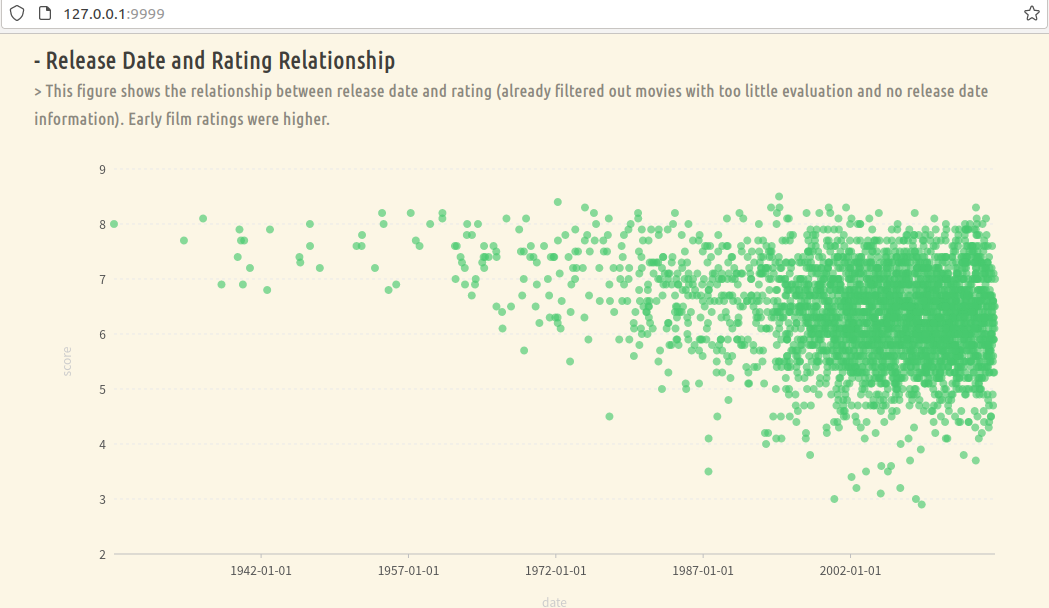
6.2关系类图表

6.2.1预算与评价的关系



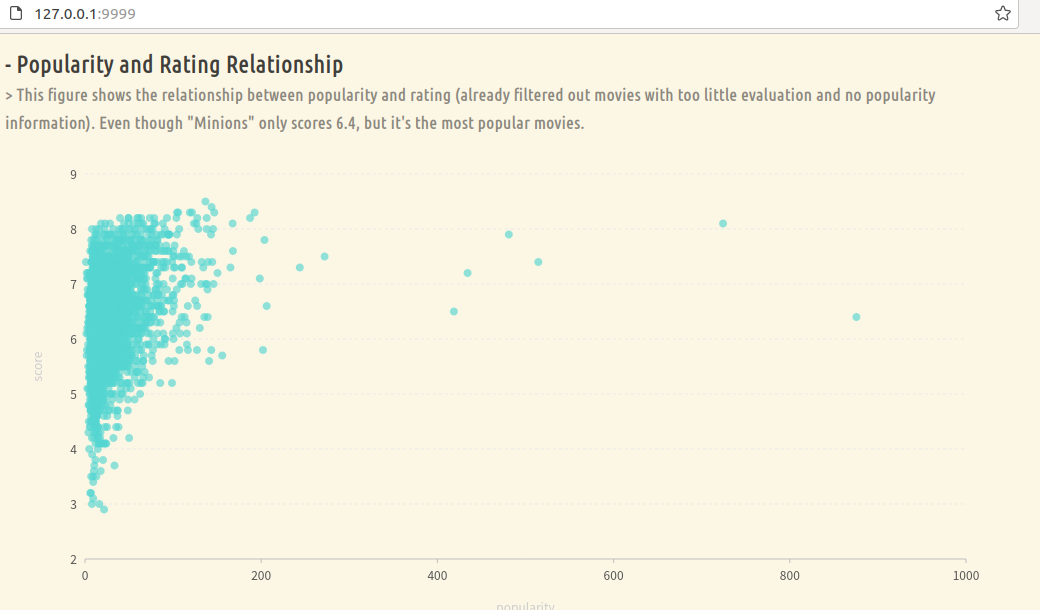
预算高的电影不见得能取得更好的评价，例如预算高达 380,000,000 美元的 Pirates of the Caribbean: On Stranger Tides（加勒比海盗）评价只有6.4分。

6.2.2发行时间与评价的关系



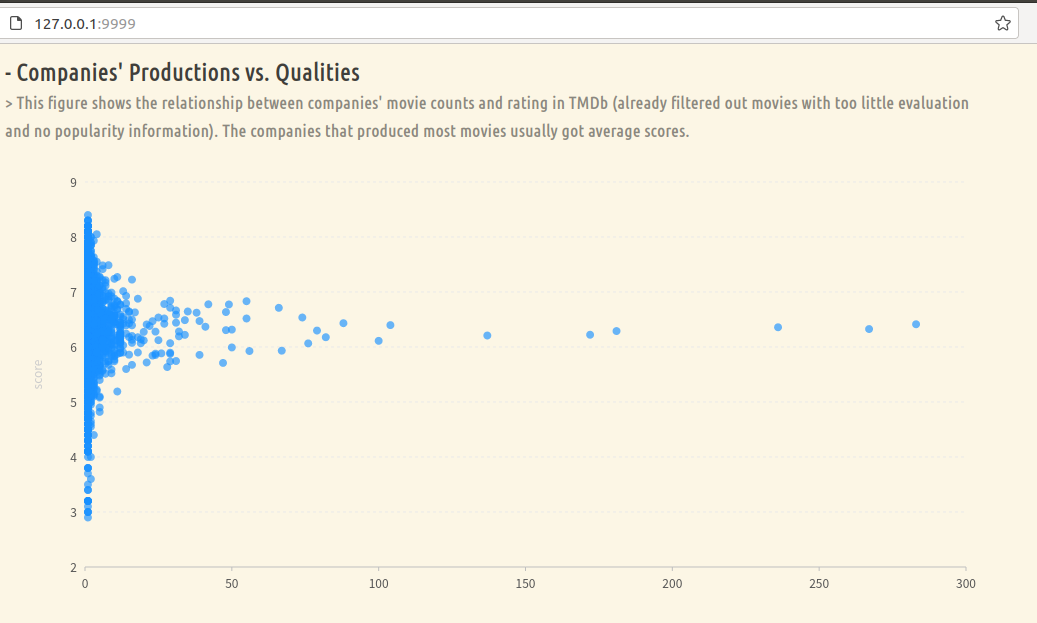
早期的电影评价都比较高，例如发行于1936年的 Modern Times（摩登时代）评价高达8.1分。

6.2.3流行度与评价的关系



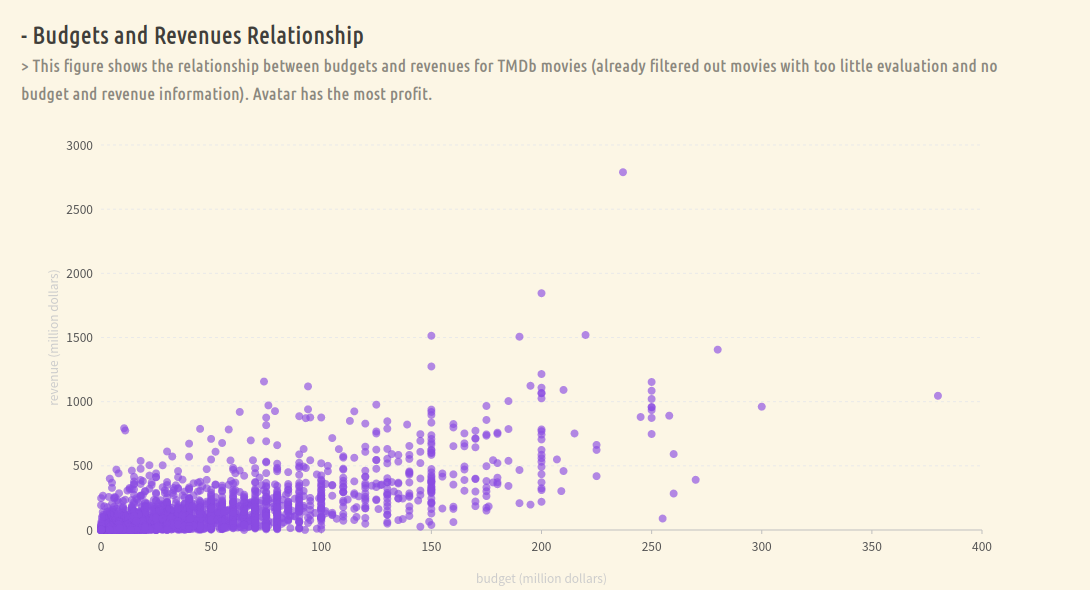
流行度较高的话一般能取得平均水平以上的评价，例如 Interstellar（星际穿越）流行度很高，评价为8.1分。

6.2.4公司生产电影平均分与数量的关系



从图中可以看出，一个公司生产的电影越多，其电影平均分越接近整体的平均水平.

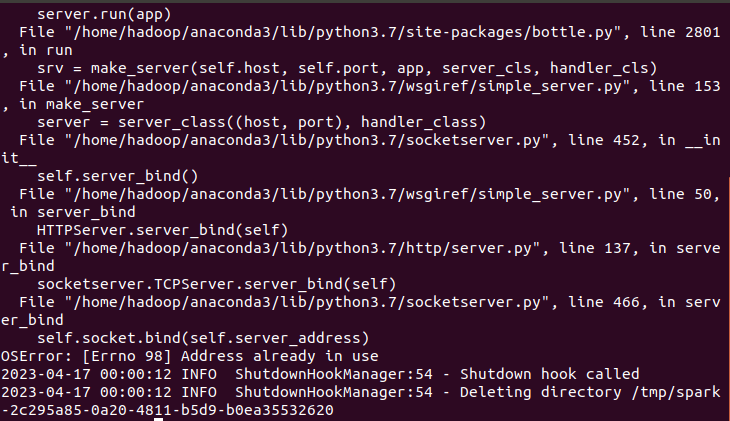
6.2.5电影预算与营收的关系



从图中可以看出，多数电影都能实现正收入，而预算为 237,000,000 美元的 Avatar（阿凡达）最终收入为2787,965,087美元.

# 二、易错点总结

## （一）使用spark-submit命令时报错



**解决方案：**该报错的原因是端口已被占用，若实在没找到使用的端口，可选择重启后再使用命令，即可正常运行



# 三、学习小记

## （一）实验心得

本次实验相比之前的难度要稍大一些，难点主要集中于对代码的理解以及对数据可视化方法的理解

环境搭建与数据预处理部分依旧没费什么力气，（因为这次试验的环境相比之前也只是需要多安装一个bottle）第一段值得关注的代码是将数据转换为DataFrame的代码，根据指南的引导，我了解到，要将数据转换为DataFrame，需要将HDFS上的数据转换为RDD，再将RDD转换为DataFrame，其大致需要经历如下过程：

·创建对象--为RDD转DataFrame制作表头--创建用于转换为DataFrame的RDD--通过 SparkSession 接口 createDataFrame ，使用准备好的表头和 RDD 创建 DataFrame

数据分析部分，即依据不同的数据构造不同的函数，再添加main函数将数据分析的过程代码进行整合，然后循环调用生成json文件，也算好理解。

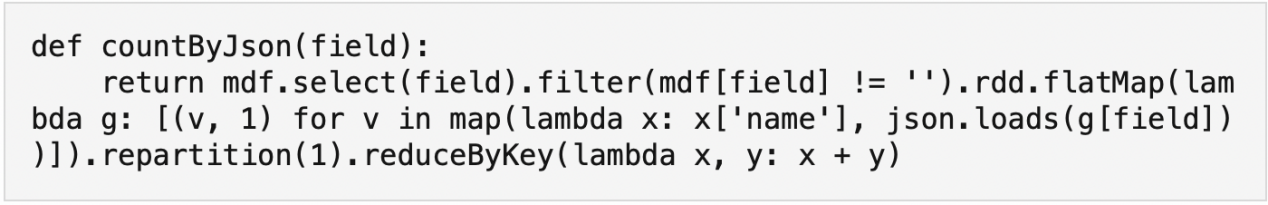
第二个难点是，运用新的方法进行可视化，由于之前的可视化基本上都是使用pyecharts实现，可以直接在jupyternotebook上生成可视化结果，而此次则使用了python web框架bottle与可视化工具G2，其大致的过程是：通过静态文件读取，使得bottle接收请求，再将 web 服务绑定到本机的 9999 端口。根据上面的实现，对于 web 页面 (html 文件)，直接放在服务启动的目录下，对于 Spark 分析的结果，则保存在 static 目录下。

总之，无论是可视化原理，还是文件转换代码，我深知自身的认知还是匮乏的，但也算是学到了些新东西，还是希望自己能够不只是局限于这次的实验，在之后的学习中也能多多运用，熟能生巧，让这些方法真正成为为我所用的工具。

## （二）数据分析代码的相关解释

2.1.TMDB电影中的体裁分布

从上面的数据字典描述可以看出，电影的体裁字段是一个 json 格式的数据，因此，为了统计不同体裁的电影的数量，需要首先解析 json 数据，从中取出每个电影对应的体裁数组，然后使用词频统计的方法统计不同体裁出现的频率，即可得到电影的体裁分布。  
首先实现一个函数 countByJson(field) ，该函数实现解析 json 格式字段从中提取出 name 并进行词频统计的功能：



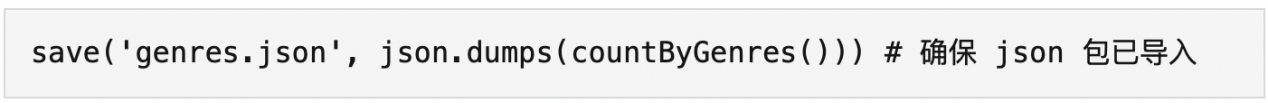
函数返回一个RDD，过程如下图所示：



基于函数 countByGenres 实现生成不同体裁的电影数统计结果：



接着，使用下面代码将数据导出至genres.json方便之后进行数据可视化：



2.2前100个常见关键词

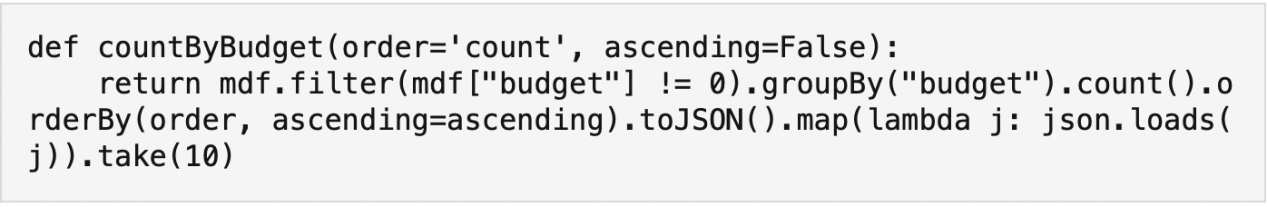
该项分析电影关键词中出现频率最高的前一百个。由于关键词字段也是 json 格式数据，因此调用 countByJson 进行频率统计，同时对于统计结果进行降序排序并取前 100 项即可：



保存依旧是使用save函数，以下便不再赘述。

2.3TMDB中最常见的十种预算数

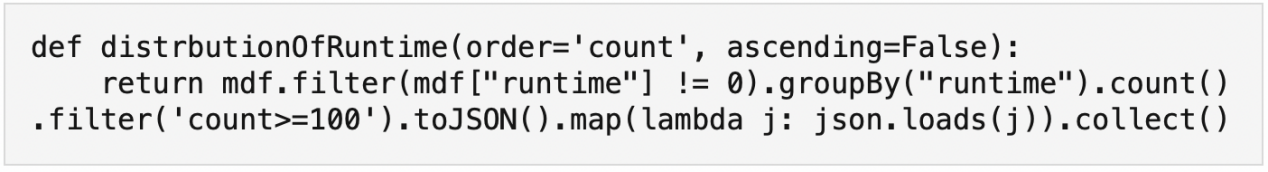
这一项探究电影常见的预算数是多少，因此需要对电影预算进行频率统计，代码如下：



首先，需要对预算字段进行过滤，去除预算为 0 的项目，然后根据预算聚合并计数，接着根据计数进行排序，并将结果导出为 json 字符串，为了统一输出，这里将 json 字符串转为 python 对象，最后取前 10 项作为最终的结果。

2.4TMDB中常见电影时长

这一项统计 TMDb 中最常见的电影时长（只显示时常>100min的电影），首先，需要过滤时长为 0 的电影，然后根据时长字段聚合并计数，接着过滤掉出现频率小于 100 的时长 （这一步是为了方便可视化，避免过多冗余信息）得到最终的结果。



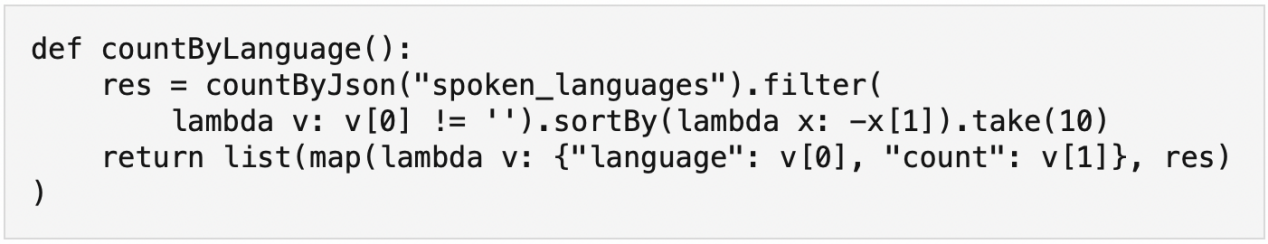
2.5生产电影最多的10大公司

这一项统计电影产出最多的 10 个公司，同样使用 countByJson 对 JSON 数据进行频率统计，然后进行降序排列取前 10 项即可。



2.6TMDB中的十大电影语言

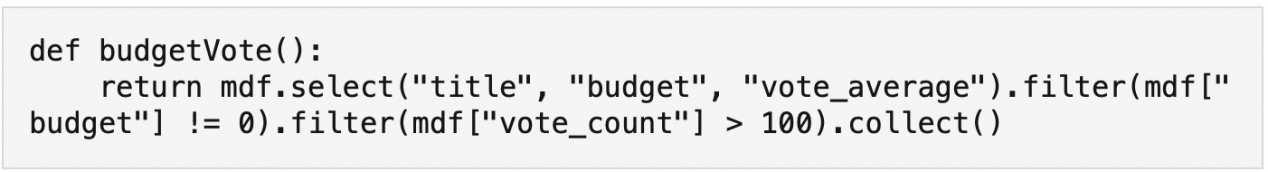
该项统计 TMDb 中出现最多的语言，与前面类似，该字段也是 JSON 数据，因此首先对每个项目进行词频统计，然后过滤掉语言为空的项目，最后排序取前十即可。



2.7预算与评价的关系

需要对每个电影导出如下数据：[电影标题，预算，评价]

基于DataFrame对数据字段进行过滤即可：



2.8发行时间与评价的关系

需要对每个电影导出如下数据：[电影标题，发行时间，评价]

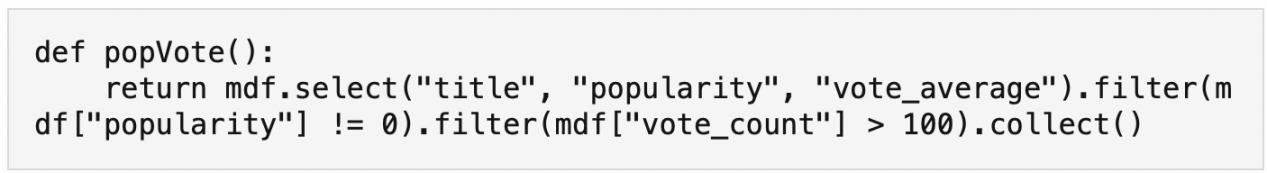
基于DataFrame对数据字段进行过滤即可：



2.9流行度与评价的关系

需要对每个电影导出如下数据：[电影标题，流行度，评价]

基于DataFrame对数据字段进行过滤即可：

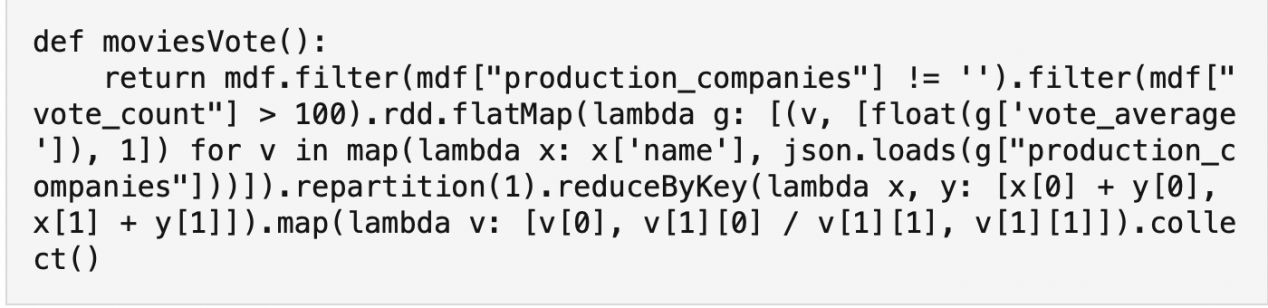


2.10公司发行电影平均分与数量的关系

需要对每个电影导出如下数据：[公司名，（评分，1）]

这部分计算每个公司生产的电影数量及这些电影的平均分分布。首先，需要对数据进行过滤，去掉生产公司字段为空和评价人数小于 100 的电影，然后对于每一条记录，得到一条如下形式的记录：

接着将所有记录的评分和计数累加，最后用总评分除以计数得到一个公司的平均评分及电影数，整个过程如下所示：



2.11电影预算和营收的关系

需要对每个电影导出如下数据：[电影标题，预算，收入]

基于DataFrame对数据字段进行过滤即可：



## （三）数据转换代码的相关解释

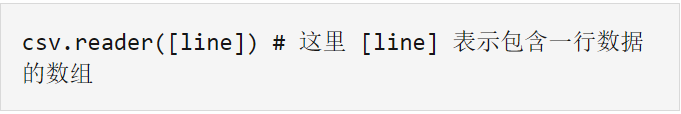
3.1文件转化

使用如下代码完成csv文件到RDD再到DataFrame的转换:

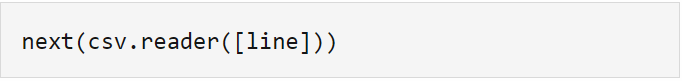


3.2创建用于转为DataFrame的RDD

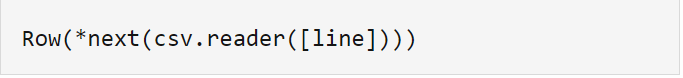
首先使用csv模块进行解析，得到一个包含每个字段的迭代器：



然后使用next函数将迭代器中的数据读取到数组中：



使用 \* 将数组转为 Row 对象的构造函数参数，创建 Row 对象：



至此，moviesRdd 中每一行为一个 Row 对象。最后，通过 SparkSession 接口 createDataFrame ，使用准备好的表头 (schema) 和 RDD 创建 DataFrame：

