# 英雄联盟大数据分析平台的构建

**作者**：付晨阳，黄为希，李嘉豪，周懿，杨磊，冯永

**单位**：重庆大学，计算机学院

**案例版权**：该案例归重庆大学计算机学院所有

**涉及的知识点**：分布式爬虫、Hadoop 、Spark、数据可视化、逻辑回归、支持向量机、朴素贝叶斯、决策树

**案例来源及案例真实性情况**：该案例来源于重庆大学计算机学院专业硕士（电子信息）课程《大数据架构与技术》中的学生精选汇编课程设计。

**摘要** 近些年来游戏产业发展十分迅速，传统的电子竞技市场也正被新兴的IT技术所推动。从调查数据来看，如今的游戏玩家数量相比于几年前的传统电竞时代已增长数倍。与此同时，随着云计算、大数据等技术在生活中的不断渗透，游戏产业也逐渐成为这些新兴技术的“沃土”。本案例以当下热门游戏“英雄联盟”为蓝本，构建英雄联盟大数据分析平台。使用爬取于英雄联盟比赛网站上的竞赛数据构建数据集，结合数据挖掘、机器学习等相关方法进行数据可视化分析进一步实现游戏胜负预测及阵容推荐等功能。本案例核心内容有：（1）使用Hadoop 和Spark搭建服务平台；（2）使用网络爬虫获取游戏比赛数据进行存储和分析；（3）进行数据可视化分析以及使用机器学习算法如逻辑回归、支持向量机、朴素贝叶斯和决策树等方法实现游戏胜负预测与阵容推荐功能。

**关键词**：分布式爬虫、Hadoop 、Spark、数据可视化、可视化分析、游戏胜负预测

## 1 引言

该教学案例来源于重庆大学计算机学院专业硕士（电子信息）课程《大数据架构与技术》中的学生精选汇编课程设计。该案例以当下热门游戏“英雄联盟”为蓝本，构建英雄联盟大数据分析平台，需引导学生进行的主要内容有：（1）掌握并使用Hadoop 和Spark搭建服务平台；（2）掌握并使用网络爬虫获取游戏比赛数据并进行存储和分析；（3）掌握数据可视化分析方法以及利用机器学习算法如逻辑回归、支持向量机、朴素贝叶斯和决策树等方法实现游戏胜负预测与阵容推荐功能。

## 2 背景介绍

近些年来游戏产业发展十分迅速，传统的电子竞技市场也正被新兴的IT技术所推动。从调查数据来看，如今的游戏玩家数量相比于几年前的传统电竞时代已增长数倍。随着当前游戏团队合作化模式的不断加深，使得很多游戏元素都出现了竞技化趋势，并且竞技强度和节奏都在逐步提高。一方面，游戏数据分析可以提升游戏竞技人员的技术。另一方面，有利于游戏公司进行游戏平衡性的改进。在挖掘出不平衡的角色之后，可以对单一数值的不同修改幅度进行情景分析。模拟不同情景下该修改对角色平衡性的影响，以此确定最佳的修改范围，作为角色技能修改的参考依据。

随着云计算、大数据等技术在生活中的不断渗透，游戏产业也逐渐成为这些新兴技术的“沃土”。因此对专业硕士进行游戏产业大数据分析的指导是有必要的。在本案例中，我们以构建英雄联盟大数据分析平台为主题，借助大数据分析与处理、数据挖掘、机器学习的相关方法，以满足培养具备创新能力与实践能力的人才的需求。

## 3 内容

该案例的主要内容主要分为三个小节，分别为服务平台的搭建、游戏数据爬取与存储以及数据可视化分析、游戏胜负预测与阵容推荐。

3.1 服务平台的搭建

3.1.1 软件版本

|  |  |
| --- | --- |
| 软件 | 版本号 |
| jdk | 1.8 |
| scala | 2.11.8 |
| hadoop | 2.7.7 |
| spark | 2.4.6 |

3.1.2节点

以三节点为例，其中一台master节点，两台slave节点

|  |  |
| --- | --- |
| 节点 | ip |
| master | 11.24.24.111 |
| slave1 | 11.24.24.112 |
| slave2 | 11.24.24.113 |

3.1.3 修改主机名和hosts

更改主机名

|  |
| --- |
| sudo vim /etc/hostname |

主节点改为master //其他两个分别改为slave1 和slave2

修改hosts

|  |
| --- |
| sudo vim /etc/hosts |

修改hosts 需要将注释的所有127.0.0.1以及locahost 取消注释！！！！不然子节点无法启动

将原文件（127.0.0.0 localhost）下面添加以下信息：

|  |
| --- |
| 11.24.24.111 master  11.24.24.112 slave1  11.24.24.113 slave2 |

保存退出后重启三台服务器

|  |
| --- |
| sudo reboot |

3.1.4 安装基础环境（Java和Scala）

对所有节点安装java和scala

Java环境搭建

下载jdk-1.8并解压到 /usr/local

|  |
| --- |
| sudo tar -zxvf jdk-8u251-linux-x64.tar.gz -C /usr/local/ |

添加java环境变量

|  |
| --- |
| sudo vim /etc/profile |

在/etc/profile中添加：

|  |
| --- |
| export JAVA\_HOME=/usr/local/java  PATH=$JAVA\_HOME/bin:$PATH  CLASSPATH=.:$JAVA\_HOME/lib/rt.jar  export JAVA\_HOME PATH CLASSPATH |

保存后刷新配置

|  |
| --- |
| source /etc/profile |

使用java -version命令确认

|  |
| --- |
| java -version |

Scala环境搭建

1、下载scala安装包scala2.13.2.tgz安装到 /usr/local/

|  |
| --- |
| sudo tar -zxvf scala2.13.2.tgz -C /usr/local/ |

2、重命名scala

|  |
| --- |
| mv scala2.13.2 scala |

3、添加Scala环境变量

|  |
| --- |
| sudo vim /etc/profile |

4、在/etc/profile中添加：

|  |
| --- |
| export SCALA\_HOME=/usr/local/scala  export PATH=$SCALA\_HOME/bin:$PATH |

5、保存后刷新

|  |
| --- |
| source /etc/profile |

6、使用scala -version命令确认

|  |
| --- |
| scala -version |

3.1.5 ssh无密码验证配置

1、对所有节点安装ssh

|  |
| --- |
| sudo yum install ssh |

2、配置master无密码登录所有slave

以下是在master节点上执行以下命令：

3、在master节点上生成密码对

|  |
| --- |
| ssh-keygen -t rsa -P '' |

Enter 继续

生成的密钥对：id\_rsa和id\_rsa.pub存储在” ~/.ssh”目录下(注意可能会不同)

把id\_rsa.pub追加到授权的key里面去

|  |
| --- |
| cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys |

修改ssh配置文件”/etc/ssh/sshd\_config”的下列内容

|  |
| --- |
| sudo vim /etc/ssh/sshd\_config |

|  |
| --- |
| RSAAuthentication yes  PubkeyAuthentication yes  AuthorizedKeysFile %h/.ssh/authorized\_keys |

重启ssh服务，才能使刚才设置有效

|  |
| --- |
| shutdown -r now |

验证无密码登录本机是否成功

|  |
| --- |
| ssh master |

注意：无密码登录成功后，使用Ctrl+D退出当前登录，防止嵌套登录

把公钥复制到所有的Slave机器上

|  |
| --- |
| scp ~/.ssh/id\_rsa.pub root@slave1:~/ #注意不能有空格  scp ~/.ssh/id\_rsa.pub root@slave2:~/ #注意直接放在主目录下 |

可以到slave节点的机器中看一下 ~/.ssh/ 是否多出了一个内容id\_rsa.pub

以下是在slave1 (11.24.24.112)节点的配置操作

将Master的公钥追加到Slave1的授权文件”authorized\_keys”中去。

|  |
| --- |
| cat ~/id\_rsa.pub >> ~/.ssh/authorized\_keys |

修改”/etc/ssh/sshd\_config”

修改ssh配置文件”/etc/ssh/sshd\_config”的下列内容

|  |
| --- |
| sudo vim /etc/ssh/sshd\_config |

|  |
| --- |
| RSAAuthentication yes  PubkeyAuthentication yes  AuthorizedKeysFile %h/.ssh/authorized\_keys |

重启ssh服务，才能使刚才设置有效

|  |
| --- |
| shutdown -r now |

用master使用ssh无密码登录slave1

|  |
| --- |
| ssh root@slave1 |

把slave节点中的” ~/”目录下的”id\_rsa.pub”文件删除掉。

|  |
| --- |
| rm -r ~/id\_rsa.pub |

重复上面的步骤把Slave2服务器进行相同的配置

配置slave无密码登录master

以下是在slave1节点(11.24.24.112)的配置操作。

创建”Slave1”自己的公钥和私钥，并把自己的公钥追加到”authorized\_keys”文件中，执行下面命令：

|  |
| --- |
| ssh-keygen -t rsa -P '' |

|  |
| --- |
| cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys |

将slave1节点的公钥”id\_rsa.pub”复制到Master节点的”/root/”目录下。

|  |
| --- |
| scp ~/.ssh/id\_rsa.pub root@master:~/ |

以下是在master节点的配置操作

将Slave1的公钥追加到Master的授权文件”authorized\_keys”中去。

|  |
| --- |
| cat ~/id\_rsa.pub >> ~/.ssh/authorized\_keys |

删除Slave1复制过来的”id\_rsa.pub”文件。

|  |
| --- |
| rm -r ~/id\_rsa.pub |

配置完成后测试从Slave1到Master无密码登录。

|  |
| --- |
| ssh master |

按照上面的步骤把Slave2和Master之间建立起无密码登录。

这样，Master能无密码验证登录每个Slave，每个Slave也能无密码验证登录到Master。

3.1.6 Hadoop完全分布式环境搭建

以下是在master节点上的操作

下载hadoop 并解压到/opt/

|  |
| --- |
| sudo tar -zxvf hadoop.tar.gz -C /opt/ |

修改环境变量配置文件

|  |
| --- |
| sudo vim /etc/profile |

添加以下内容

|  |
| --- |
| export HADOOP\_HOME=/opt/hadoop  export PATH=$PATH:$HADOOP\_HOME/bin  export PATH=$PATH:$HADOOP\_HOME/sbin  export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME  export HADOOP\_COMMON\_HOME=$HADOOP\_HOME  export HADOOP\_HDFS\_HOME=$HADOOP\_HOME  export YARN\_HOME=$HADOOP\_HOME  export HADOOP\_ROOT\_LOGGER=INFO,console  export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native  export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib" |

修改完成后执行

|  |
| --- |
| source /etc/profile |

修改hadoop的相关配置

进入hadoop安装路径，执行

|  |
| --- |
| sudo vim ./etc/hadoop/hadoop-env.sh |

指定JAVA\_HOME

|  |
| --- |
| export JAVA\_HOME=/usr/local/java |

修改$HADOOP\_HOME/etc/hadoop/slaves，将原来的localhost删除，改成如下内容：

|  |
| --- |
| slave1  slave2 |

修改HADOOP\_HOME/etc/hadoop/core-site.xml

|  |
| --- |
| <configuration>  <property>  <name>fs.defaultFS</name>  <value>hdfs://master:9000</value>  </property>  <property>  <name>io.file.buffer.size</name>  <value>131072</value>  </property>  <property>  <name>hadoop.tmp.dir</name>  <value>/opt/hadoop/tmp</value>  </property>  </configuration> |

修改$HADOOP\_HOME/etc/hadoop/hdfs-site.xml

|  |
| --- |
| <configuration>  <property>  <name>dfs.namenode.secondary.http-address</name>  <value>master:50090</value>  </property>  <property>  <name>dfs.replication</name>  <value>2</value>  </property>  <property>  <name>dfs.namenode.name.dir</name>  <value>file:/opt/hadoop/hdfs/name</value>  </property>  <property>  <name>dfs.datanode.data.dir</name>  <value>file:/opt/hadoop/hdfs/data</value>  </property>  </configuration> |
|  |

复制template，生成xml，命令如下：

|  |
| --- |
| cp etc/hadoop/mapred-site.xml.template etc/hadoop/mapred-site.xml |

修改$HADOOP\_HOME/etc/hadoop/mapred-site.xml

|  |
| --- |
| <configuration>  <property>  <name>mapreduce.framework.name</name>  <value>yarn</value>  </property>  <property>  <name>mapreduce.jobhistory.address</name>  <value>master:10020</value>  </property>  <property>  <name>mapreduce.jobhistory.address</name>  <value>master:19888</value>  </property>  </configuration> |

修改$HADOOP\_HOME/etc/hadoop/yarn-site.xml

|  |
| --- |
| <configuration>  <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>  <property>  <name>yarn.resourcemanager.address</name>  <value>master:8032</value>  </property>  <property>  <name>yarn.resourcemanager.scheduler.address</name>  <value>master:8030</value>  </property>  <property>  <name>yarn.resourcemanager.resource-tracker.address</name>  <value>master:8031</value>  </property>  <property>  <name>yarn.resourcemanager.admin.address</name>  <value>master:8033</value>  </property>  <property>  <name>yarn.resourcemanager.webapp.address</name>  <value>master:8088</value>  </property>  </configuration> |

复制master节点的hadoop文件夹到slave1和slave2上

|  |
| --- |
| scp -r /opt/hadoop root@slave1:/opt |

在Slave1和Slave2上分别修改/etc/profile，(hadoop的配置信息) 过程同master一样

格式化HDFS（只有首次部署才可以使用，谨慎操作，只在master上操作）

|  |
| --- |
| hadoop namenode -format |

启动集群（只在master操作）

|  |
| --- |
| ./opt/hadoop/sbin/start-all.sh |

查看三台集群是否启动成功

|  |
| --- |
| jsp |

3.1.7 Spark完全分布式环境搭建

以下操作都是在master节点进行

1) 下载二进制包spark-3.0.0-bin-hadoop2.7.tgz

2) 解压到相应目录

|  |
| --- |
| tar -zxvf spark-3.0.0-bin-hadoop2.7.tgz -C /opt |
| mv /opt/spark-3.0.0-bin-hadoop2.7.tgz /opt/spark |

3) 修改相应的配置文件

修改/etc/profie，增加如下内容：

|  |
| --- |
| export SPARK\_HOME=/opt/spark/  export PATH=$PATH:$SPARK\_HOME/bin |

复制spark-env.sh.template成spark-env.sh

|  |
| --- |
| sudo cp conf/spark-env.sh.template conf/spark-env.sh |

修改$SPARK\_HOME/conf/spark-env.sh，添加如下内容.（标注需要改为自己的版本）

|  |
| --- |
| export JAVA\_HOME=/usr/local/java  export SCALA\_HOME=/usr/local/scala  export HADOOP\_HOME=/opt/hadoop  export HADOOP\_CONF\_DIR=/opt/hadoop/etc/hadoop  export SPARK\_MASTER\_IP=11.24.24.111  export SPARK\_MASTER\_HOST=11.24.24.111  #export SPARK\_LOCAL\_IP=11.24.24.112  #export SPARK\_LOCAL\_IP=11.24.24.113  export SPARK\_LOCAL\_IP=0.0.0.0  export SPARK\_WORKER\_MEMORY=1g  export SPARK\_WORKER\_CORES=2  export SPARK\_HOME=/opt/spark  export SPARK\_DIST\_CLASSPATH=$(/opt/hadoop/bin/hadoop classpath) |

复制slaves.template成slaves

|  |
| --- |
| sudo cp conf/slaves.template conf/slaves |

修改$SPARK\_HOME/conf/slaves，在原来的localhost下面改为如下内容：

|  |
| --- |
| master  slave1  slave2 |

（注意，如果不删除之前的localhost，spark搭建完成之后，会出现多个worker）

4) 将配置好的spark文件复制到slave1和slave2节点

|  |
| --- |
| scp -r /opt/spark root@slave1:~ //拷贝整个目录  scp -r /opt/spark root@slave2:~  sudo mv spark//opt  sudo mv spark//opt |

5) 修改slave1和slave2配置。

root@slave1:~$ sudo vim /etc/profile

在slave1和slave2上分别修改/etc/profile，增加Spark的配置，过程同master一样。

|  |
| --- |
| export SPARK\_HOME=/opt/spark/  export PATH=$PATH:$SPARK\_HOME/bin |

在slave1和slave2修改$SPARK\_HOME/conf/spark-env.sh

root@master: $ sudo vim /opt/spark-2.1/conf/spark-env.sh

将export SPARK\_LOCAL\_IP=192.168.110.XXX改成slave1和slave2对应节点的IP

6) 在Master节点启动集群

先启动hadoop，再启动spark

root@master:/opt$ /opt/hadoop/sbin/start-all.sh

root@master:/opt$ /opt/spark/sbin/start-all.sh

查看集群是否启动成功：jps

3.2 游戏数据爬取与存储

3.2.1 分布式爬虫环境搭建

为了运行分布式爬虫，首先应该搭建完整的环境。

通过使用pip指令安装scrapy就能完成scrapy的搭建。

|  |
| --- |
| pip install scrapy |

通过下载redis并在centos上进行解压安装，并自行构建配置文件以完成redis的搭建。

|  |
| --- |
| wget <http://download.redis.io/releases/redis-x.x.xx.tar.gz>  tar -xzvf redis-x.x.xx.tar.gz  cd redis-x.x.xx  make  cd src  make install |

同时，为了防止数据库受到攻击，所以在配置文件中新增password选项并配置redis数据库密码以及更改其他可选项。

|  |
| --- |
| cd ../  vim redis.conf  password xxxxxx  daemonize yes |

由于scrapy使用python运行，因此还需要使用pip指令安装用于操作数据库的redis包和用于实现分布式爬虫的scrapy-redis包。

|  |
| --- |
| pip install redis  pip install scrapy-redis |

由于一场英雄联盟比赛中发生的事件数量不能确定，因此我们项目的数据不适合存放于关系型数据库中，所以我们使用mongodb进行数据的存储。通过下载mongodb并在centos上进行解压安装，并自行构建配置文件以完成mongodb的搭建。

|  |
| --- |
| wget https://fastdl.mongodb.org/linux/mongodb-linux-x86\_64-ubuntu1604-x.x.xx.tgz  tar -xzvf mongodb-linux-x86\_64-ubuntu1604-x.x.xx.tgz  cd mongodb-linux-x86\_64-ubuntu1604-x.x.xx  mkdir data  mkdir logs  cd logs  touch mongo.log |

配置mongodb的服务配置，以便设置mongodb的运行模式。

|  |
| --- |
| dbpath=/mongodb/data  logpath=/mongodb/logs/mongo.log  logappend=true  journal=true  quiet=true  port=27017  fork=true  bind\_ip=0.0.0.0  auth=false |

为了在使用python是能够对mongodb进行读取与存储，因此，还要安装mongodb相关的python包。

|  |
| --- |
| pip install pymongo |

3.2.2 数据收集

在网站<https://www.op.gg/>上存在大量的英雄联盟比赛的相关数据，因此我们的爬虫目标主要为该网站上的各场比赛。而我们的爬虫程序是一种基于scrapy-redis的分布式爬虫程序，这种模式允许多台子机同时进行爬虫程序的运行并保证爬取的信息互不相同，与单机爬虫程序相比可以更好地利用服务器集群的数量优势。

在英雄数据界面，我们先使用了一个单机爬虫ChampionSpider爬取了各个英雄的相关数据并将之存入远程服务器的数据库中，存储为表Champion，数据结构与功能详见1.3。

然后，在排行榜中，我们又使用另一个单机爬虫SummonerSpider爬取了排行榜中段位为master及其以上的所有玩家并将之存入远程服务器的数据库中，存储为表Summoner，数据结构与功能详见1.3。

最后，通过表Summoner中的相关数据进行分布式爬虫爬取每位Summoner最近的20场比赛。通过使用文件SpiderStart.py将表Summoner中的数据放入新表Summoner \_scrap中用于召唤师url数据的暂存以实现分布式爬虫的随时中断与继续运行（该文件同样可以用于重置召唤师的url数据）。分布式爬虫MatchSpider从表Summoner\_scrap中提取url并进入该网页爬取该召唤师的最新的20场比赛并将之存入远程服务器的数据库中，存储为表Match。分布式爬虫每次发放16个url给每个子机，子机在运行的同时边将爬取到的数据放入主机与子机之间的管道，进行阻塞式的数据存放。当所有子机均停止运行时，主机不再分发任务，而当有一架子机开始运行时，主机便能从上次所有子机中断的位置继续分发任务。同时，由于在爬取Match相关信息时发现每场比赛的数据使用动态网页技术，因此还需要构造新的url来实现完整信息的爬取并对Match进行补充，动态网页1需要召唤师的id、比赛的id和moreload标志，动态网页2需要召唤师id和比赛的id。数据结构与功能详见1.3。

3.2.3 数据结构与格式

表Champion中包含了英雄联盟中英雄的主要信息，包括name（英雄名称），lane（五种不同的游戏分工），url（头像图片的url）。该表用于进行阵容推荐、可视化展示和一些数据的统计。

表Summoner和Summoner\_scrap中包含了英雄联盟中各用户的主要信息，包括name（用户名），url（用户统计界面url）。该前者用于进行用户的去重而后者用于爬虫进度的监管和维护，保障爬虫程序的正常停止与续行，允许任意数量的子机可以在任何时间参与爬虫。

表Match较为复杂，其中包含了英雄联盟比赛的相关信息。该表用于进行胜率预测和大部分的数据统计。表结构如下。

Match表收集的数据存入mongodb便能够很方便地使用shell脚本，进行数据的格式的转换并输出json文件再将文件部署至搭建好的hdfs系统中以便spark算子调用。

|  |
| --- |
| mongoexport -d LOL -c Match -o Match.json  hdfs dfs -put Match.json /Match.json |

3.3 数据可视化分析、胜负预测与阵容推荐

本项目主要实现以下四个功能并进行可视化展示：

1. 使用echarts制作了多组统计图表，对爬取到的数据进行展示。
2. 使用逻辑斯蒂回归、支持向量机分类、决策树、朴素贝叶斯等模型进行比赛胜负预测。
3. 计算各因素之间的皮尔逊相关系数，分析获胜因素。
4. 根据获胜相关指数针对用户需求进行阵容推荐。

如图1及图2为可视化界面的整体布局。界面从左至右可分为三个部分，第一和第三部分展示的是统计图表以及比赛胜负预测模型的效果。中间部分可以点击标签在两个模块——热力图和阵容推荐之间进行切换。

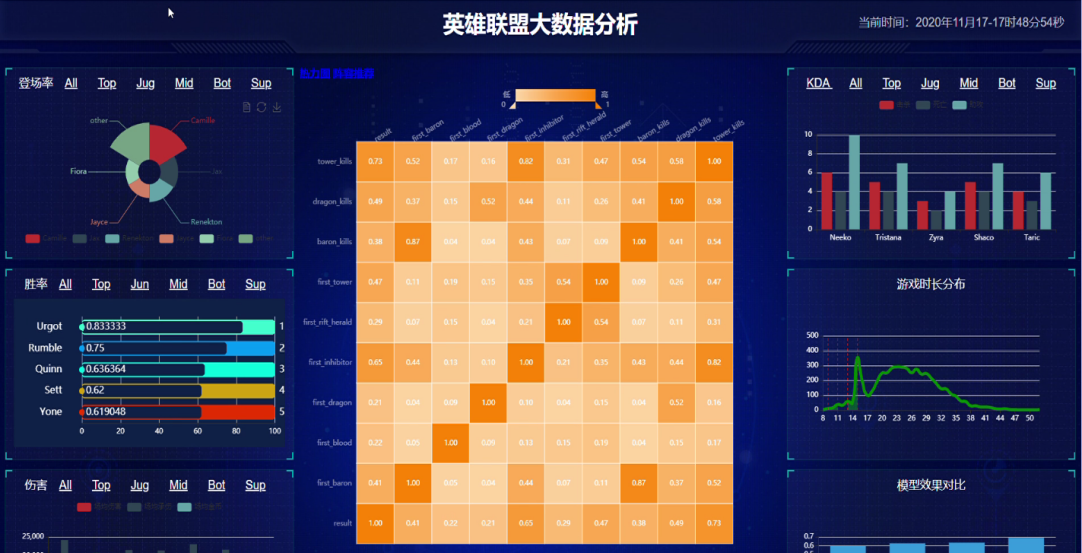


图1



图2

3.3.1 springboot

Spring框架是Java平台上的一种开源应用框架，提供具有控制反转特性的容器。Spring框架为开发提供了一系列的解决方案，比如利用控制反转的核心特性，并通过依赖注入实现控制反转来实现管理对象生命周期容器化，利用面向切面编程进行声明式的事务管理，整合多种持久化技术管理数据访问，提供大量优秀的Web框架方便开发等等。Spring框架具有控制反转（IOC）特性，IOC旨在方便项目维护和测试，它提供了一种通过Java的反射机制对Java对象进行统一的配置和管理的方法。Spring框架利用容器管理对象的生命周期，容器可以通过扫描XML文件或类上特定Java注解来配置对象，开发者可以通过依赖查找或依赖注入来获得对象。Spring框架具有[面向切面编程](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%88%87%E9%9D%A2%E7%BC%96%E7%A8%8B/6016335" \t "https://baike.baidu.com/item/Spring%20Boot/_blank)（AOP）框架，SpringAOP框架基于代理模式，同时运行时可配置；AOP框架主要针对模块之间的交叉关注点进行模块化。Spring框架的AOP框架仅提供基本的AOP特性，虽无法与AspectJ框架相比，但通过与AspectJ的集成，也可以满足基本需求。Spring框架下的事务管理、远程访问等功能均可以通过使用SpringAOP技术实现。Spring的事务管理框架为Java平台带来了一种抽象机制，使本地和全局事务以及嵌套事务能够与保存点一起工作，并且几乎可以在Java平台的任何环境中工作。Spring集成多种事务模板，系统可以通过事务模板、XML或Java注解进行事务配置，并且事务框架集成了消息传递和缓存等功能。Spring的数据访问框架解决了开发人员在应用程序中使用数据库时遇到的常见困难。它不仅对Java:JDBC、iBATS/MyBATIs、Hibernate、Java数据对象（JDO）、ApacheOJB和ApacheCayne等所有流行的数据访问框架中提供支持，同时还可以与Spring的事务管理一起使用，为数据访问提供了灵活的抽象。

SpringBoot所具备的特征有：

（1）可以创建独立的Spring应用程序，并且基于其Maven或Gradle插件，可以创建可执行的JARs和WARs；

（2）内嵌Tomcat或Jetty等Servlet容器；

（3）提供自动配置的“starter”项目对象模型（POMS）以简化Maven配置；

（4）尽可能自动配置Spring容器；

（5）提供准备好的特性，如指标、健康检查和外部化配置；

（6）绝对没有代码生成，不需要XML配置。

3.3.2 Echarts

ECharts 是一个使用 JavaScript 实现的开源可视化库，涵盖各行业图表，满足各种需求。提供了常规的折线图、柱状图、散点图、饼图、K线图，用于统计的盒形图，用于地理数据可视化的地图、热力图、线图，用于关系数据可视化的关系图、treemap、旭日图，多维数据可视化的平行坐标，还有用于 BI 的漏斗图，仪表盘，并且支持图与图之间的混搭。其中内置的 dataset 属性（4.0+）支持直接传入包括二维表，key-value 等多种格式的数据源，此外还支持输入 TypedArray格式的数据。通过增量渲染技术（4.0+），配合各种细致的优化，ECharts 能够展现千万级的数据量。Echarts还针对移动端交互做了细致的优化，例如移动端小屏上适于用手指在坐标系中进行缩放、平移。 PC 端也可以用鼠标在图中进行缩放（用鼠标滚轮）、平移等。

3.3.3统计图表

1. 登场率

分别统计全体英雄登场率以及各分路英雄登场率，并使用饼图展示出全体英雄登场率以及各分路英雄登场率的前五名。如图3所示，上路登场率前五名分别是Camille,Jax,Renekton,Jayce,Fiora。

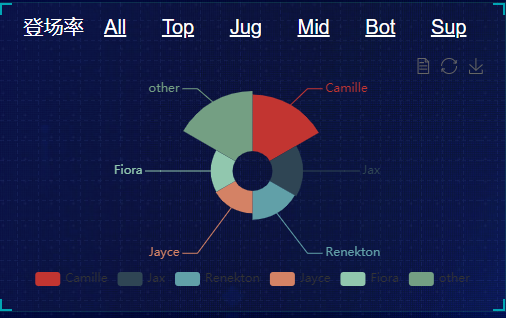


图3

2. 胜率

分别统计全体英雄胜率以及各分路英雄胜率，并使用柱状图展示出全体英雄胜率以及各分路英雄胜率的前五名。如图4所示，下路胜率前五名分别是Jinx,Lucian,Samira,Jhin,Yasuo。

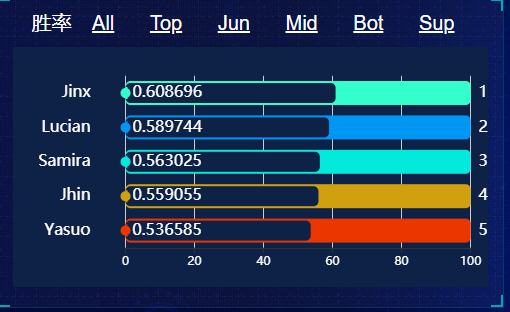


图4

3. 伤害、承伤、金币

分别统计全体英雄以及各分路英雄伤害、承伤、金币。如图5所示，可点击All，Top，Jug等标签切换分路，也可点击图例单独查看伤害、承伤或者金币。

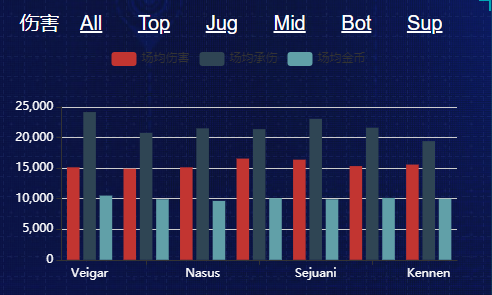


图5

4. KDA

分别统计全体英雄以及各分路英雄KDA。如图6所示，可点击All，Top，Jug等标签切换分路，也可点击图例单独查看击杀数，死亡数或者助攻数。

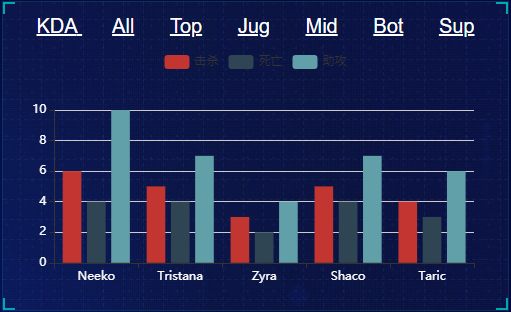


图6

5. 游戏时长

统计所有比赛的时长并画出折线图。如图7所示，游戏最常在15分钟左右结束，这应该与游戏最早投降时间是15分钟这一设置有关。

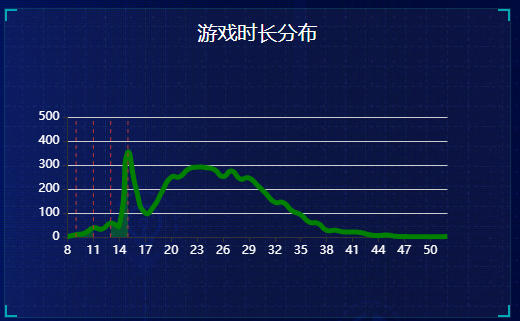


图8

3.3.4 胜负预测

首先提取比赛中双方英雄数据并对英雄进行one-hot编码，然后使用主成分分析进行降维，最后采用逻辑斯蒂回归、支持向量机分类、决策树、朴素贝叶斯等模型进行预测。如图4.8所示，朴素贝叶斯模型的预测准确率最高，达到70%左右。

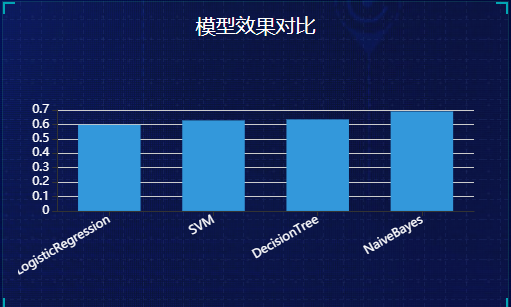


图8

3.3.5获胜因素分析

首先提取比赛数据中各资源数据，然后计算各特征之间的皮尔逊相关系数并用热力图进行可视化展示。如图9所示，热力图中横纵坐标列出比赛中各因素，网格中数字越大，颜色越深代表两者之间的相关性越强。

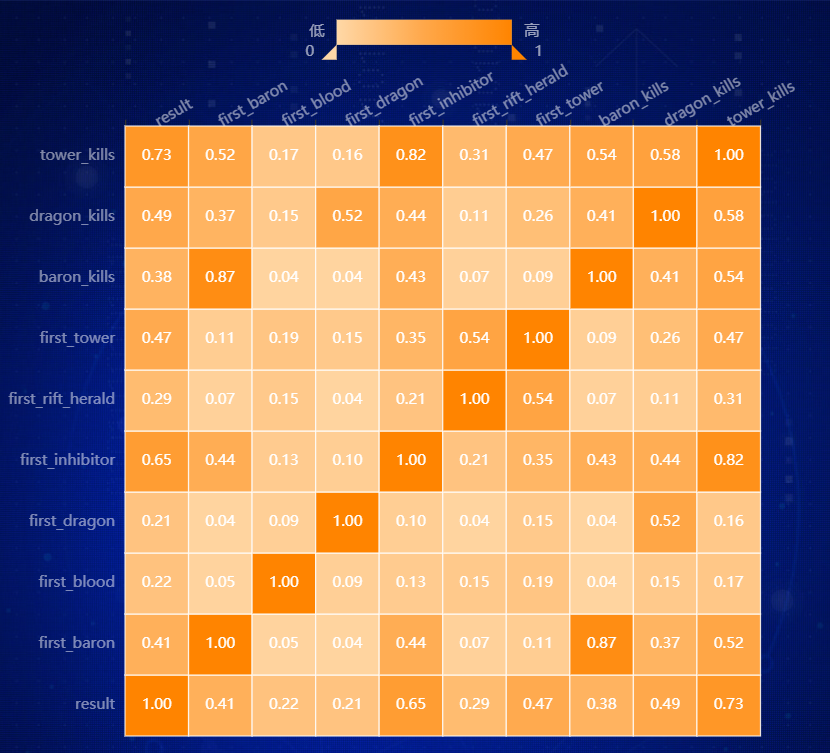


图9

3.3.6 阵容推荐

首先预处理比赛数据，提取比赛阵容中任意两个英雄间的获胜相关指数。根据获胜相关指数及英雄所处位置，为每个位置选取一个英雄，计算总的获胜相关指数。再依据总的获胜相关指数大小，进行阵容推荐。如图10所示，用户依次选择了四个英雄，系统会推荐包含用户已选英雄的阵容。



图10

## 4 小结

本案例以当下热门游戏“英雄联盟”为蓝本，构建英雄联盟大数据分析平台。使用爬取于英雄联盟比赛网站上的竞赛数据构建数据集，结合数据挖掘、机器学习等相关方法进行数据可视化分析进一步实现游戏胜负预测及阵容推荐等功能。主要内容包括：（1）使用Hadoop 和Spark搭建服务平台；（2）使用网络爬虫获取游戏比赛数据进行存储和分析；（3）进行数据可视化分析以及使用机器学习算法如逻辑回归、支持向量机、朴素贝叶斯和决策树等方法实现游戏胜负预测与阵容推荐功能。该案例主题新颖，结合了工业界现实需求与大数据分析与挖掘的多种理论与技术，可以充分增强学生的实践能力与理论基础。另外，本案例的内容仅为指导性的过程，在实际教学中，可保持基本研究内容不变，鼓励学生引入其它的数据预处理、数据挖掘、机器学习方法完成任务。

## 附录

1. 本案例提供配套的PPT、视频、数据集与代码等，发布于Github，链接为：https://github.com/Wanghui-Huang/CQU\_bigdata。

2. 本案例涉及到数据预处理以及多种机器学习算法，建议使用python语言进行编写，推荐的工具包有requests（网络请求库），BeautifulSoup（HTML解析库），scikit-learn（机器学习库）。

3. 本案例参考文献如下：

[1] 周志华. 机器学习[M]. 北京: 清华大学出版社, 2016.

[2] Goodfellow I, Bengio Y, Courville A. Deep learning[M]. MIT press, 2016.