# 基于Spark的实时攻击检测

**作者**：黄洪，凌诗钧，李海峰，刘力，李亚玲，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**：redis数据库设计、flume+kafka环境搭建、SparkStreaming流式实时处理、决策树回归、深度神经网络、数据可视化

**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** 分布式拒绝服务攻击可以在同一时间对多台计算机进行攻击，从而使被攻击的目标无法正常使用，造成巨大的经济损失。同时分布式拒绝服务攻击方式在进行攻击的时候，可以对源IP地址进行伪造，因此这种攻击方式成为了非常难以防范的攻击。随着大数据与机器学习技术的不断发展，如何利用一定的算法或数据处理技术进行攻击的实时检测逐渐成为一个热点问题。本案例以基于Spark的实时攻击检测为主题，基于Hadoop&Spark集群环境，通过对入侵检测数据集进行建模和选取，从而对分布式拒绝服务攻击进行实时检测并对结果进行可视。在具体实现上首先基于Flume+Kafka架构，利用SparkStreaming流式实时处理技术，完成日志实时解析的目标。然后构建了决策树和深度学习预测模型，选取结果较优的模型作为最终的检测模型。最后利用SparkStreaming进行实时检测并对结果进行了可视化。

**关键词**：SparkStreaming流式实时处理，决策树回归，深度学习，数据可视化

## 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。该案例的关键问题为基于Spark的实时攻击检测，需引导学生进行的主要内容有：（1）搭建Flume+Kafka架构，连接Spark Streaming流式实时处理技术，完成日志实时解析的目标；（2）构建决策树和深度学习模型，选取更优的模型作为最终的检测模型对攻击进行检测；（3）Spark Streaming实时检测并对结果进行可视化。

## 2 背景介绍

分布式拒绝服务攻击可以在同一时间对多台计算机进行攻击，从而使被攻击的目标无法正常使用。对于大型网站而言，分布式拒绝服务攻击会造成网站无法正常运行，这样不仅仅会影响用户的正常使用，同时会造成巨大的经济损失。分布式拒绝服务攻击方式在进行攻击的时候，可以对源IP地址进行伪造，这就使得这种攻击在发生的时候具有极高的隐蔽性，同时要对攻击进行检测也是非常困难的，因此这种攻击方式也成为了非常难以防范的攻击。目前来说不仅是对DDoS，更是对所有网络的攻击，都应采取尽可能周密的防御措施，同时加强对系统的检测，建立迅速有效的应对策略。可采取的防御措施包括：

(1)全面综合地设计网络的安全体系，注意所使用的安全产品和网络设备。

(2)提高网络管理人员的素质，关注安全信息，遵从有关安全措施，及时地升级系统，加强系统抗击攻击的能力。

(3)在系统中加装防火墙系统，利用防火墙系统对所有出入的数据包进行过滤，检查边界安全规则，确保输出的包受到正确限制。

(4)优化路由及网络结构。对路由器进行合理设置，降低攻击的可能性。

(5)优化对外提供服务的主机，对所有在网上提供公开服务的主机都加以限制。

(6)安装入侵检测工具(如NIPC、NGREP)，经常扫描检查系统，解决系统的漏洞，对系统文件和应用程序进行加密，并定期检查这些文件的变化。

然而这些方法都依赖于设备与人员素质的提升。在这种前提下，将大数据与机器学习结合，开发一种基于Spark的实时攻击检测系统就具有极大的价值。

## 3 内容

该案例的主要内容主要分为三个小节，分别为hadoop+spark集群部署、模型搭建训练，以及SparkStreaming实时检测。

**3.1 hadoop+spark集群部署**

hadoop+spark集群，分别包含一台master主机与一台work节点。

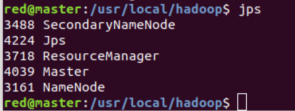
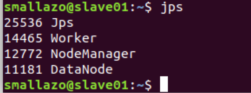
 

Fig1.Master节点 Fig2.work节点

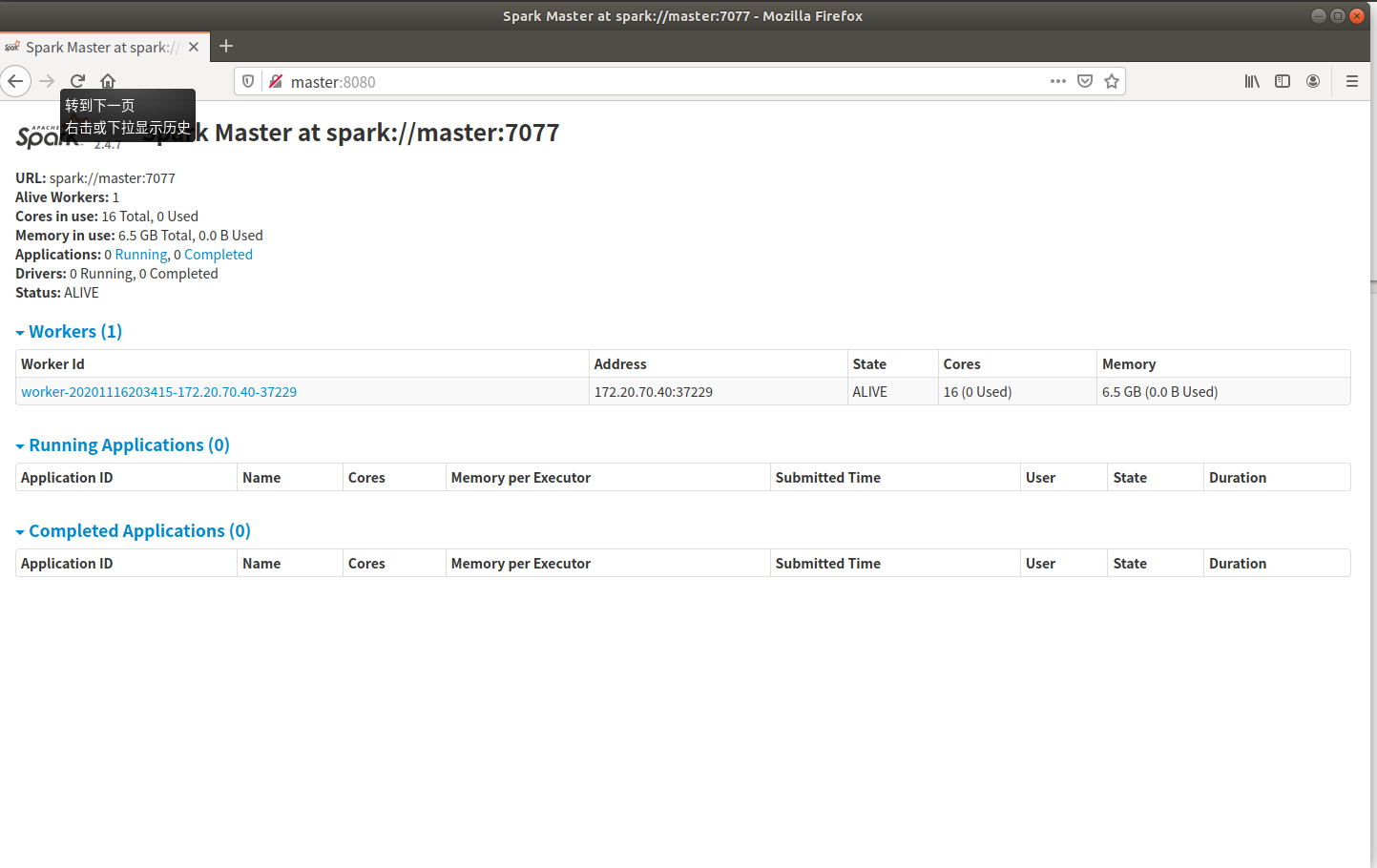


Fig3.WebUI



Fig4.伪分布式

**3.1.2 flume+kafka**

由于资源有限，可能不能部署flume+kafka的集群，在一台主机上配置了flume+kafka，部署方式相比hadoop与spark集群要简单的多，下载、解压好对应的版本后，我们只需要修改对应的配置。

**1) flume配置**

|  |
| --- |
| **c**d /usr/local/flume //前提是下载解压到相应的位置  cp ./conf/flume-conf.properties.template ./conf/flume-conf  //修改flume-conf文件如下 |

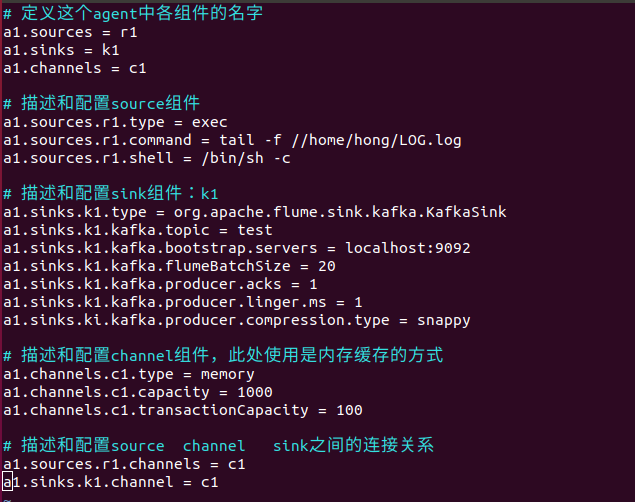
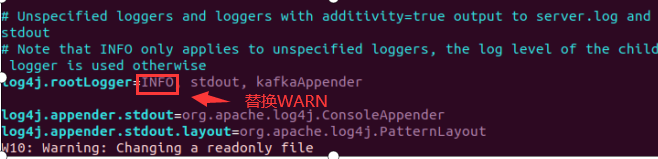


Fig5.flume配置

**2) kafka配置**

|  |
| --- |
| cd /usr/local/kafka/config  vim log4j.properties //修改日志文件，禁止INFO输出，修改如图  vim server.properties //打开监听端口,修改如图 |



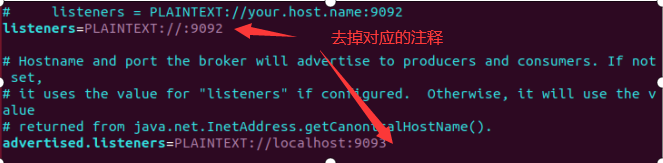


Fig6.kafka配置

**3.1.3 java+python**

Ubuntu18.04安装java

|  |
| --- |
| sudo apt install openjdk-8-jdk-headless |

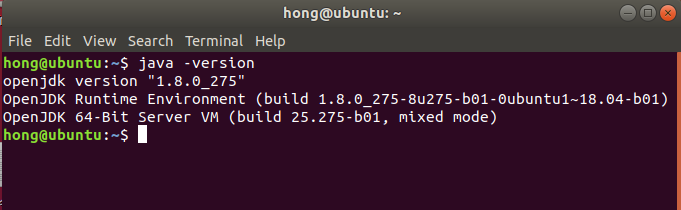


Fig7.java版本

Ubuntu18.04自带python3的版本无需安装其他版本

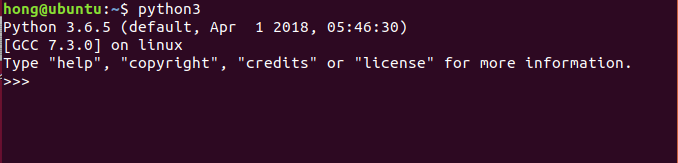


Fig8.python版本

**3.1.4 MySQL+Redis**

Ubuntu18.04安装MySQL

|  |
| --- |
| sudo apt install mysql-server //安装mysql  netstat -tap | grep mysql //查看是否安装成功  mysql -u root -p //登录mysql |



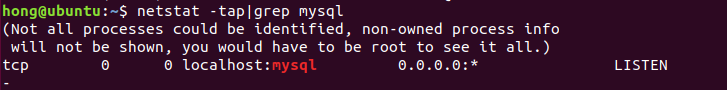


Fig9.mysql配置

Ubuntu18.04安装Redis

|  |
| --- |
| sudo apt-get install redis-server //安装redis  redis-cli //启动redis客户端 |





Fig10.redis配置

**3.1.5 python模块**

|  |
| --- |
| **pip3 install redis //用于操作redis**  pip3 install MySQL-python //用于操作mysql  pip3 install matplotlib //用于可视化展示 |

**3.1.6 kafka作为spark高级输入源需要的库**

kafka作为高级输入源，需要依赖独立的库（jar文件）。已经安装好的Spark版本，这些jar包都不在其中，需要另外下载，这里需要下载spark-streaming-kafka-0-8\_2.11-2.4.0.jar。其他版本下载路径可以到MVNREPOSITORY官网。

|  |
| --- |
| <https://repo1.maven.org/maven2/org/apache/spark/spark-streaming-kafka-0-8_2.11/2.4.7/> |

需要把这个文件复制到Spark目录的jar目录下

|  |
| --- |
| cp ./spark-streaming-kafka-0-8\_2.11-2.4.0.jar /usr/local/spark/jars |

**3.1.7 pyspark词频统计测试**

读取本地spark自带的README.md文件进行词频统计

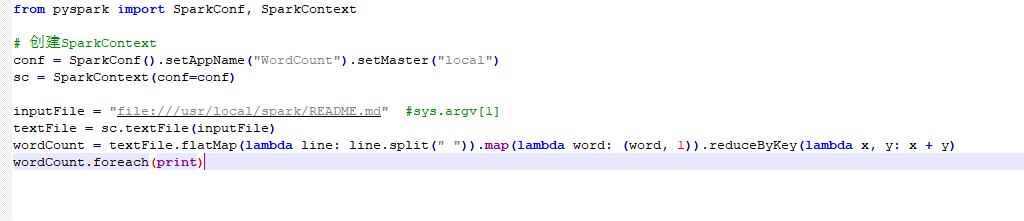


Fig11.pyspark词频统计源码



Fig12.pyspark词频统计结果

**3.1.8 flume+kafka 测试**

Flume与kafka启动命令

|  |
| --- |
| ##启动ZooKeeper  cd /usr/local/kafka  bin/zookeeper-server-start.sh config/zookeeper.properties  ##启动Kafka  bin/kafka-server-start.sh config/server.properties  ##创建 Topic  bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test  ##启动一个消费者  bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning  ##启动flume  cd /usr/local/flume  bin/flume-ng agent --conf conf --conf-file ./conf/flume-conf --name a1 -Dflume.root.logger=INFO,console |

通过上述命令启动flume与kafka之后就可以通过向flume监控的日志文件写数据，在kafka消费者查看新增的数据。

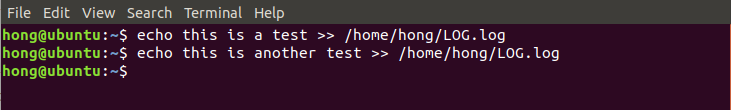


Fig13.向日志文件写入数据

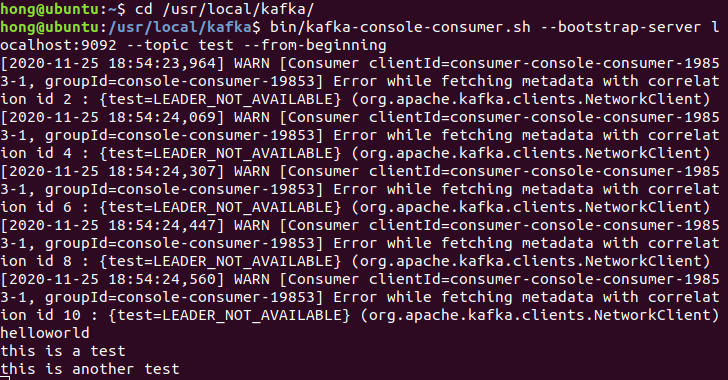


Fig14.kafka消费者

**3.1.9 SparkStreaming+kafka+flume 实时词频统计**

向spark提交任务，然后向flume监控的日志文件写数据，SparkStreaming读取kafka中的数据进行词频统计实时输入到控制台。

|  |
| --- |
| /usr/local/spark/bin/spark-submit test2.py localhost:2181 test ###test是topic，localhost:2181是kafka源 |

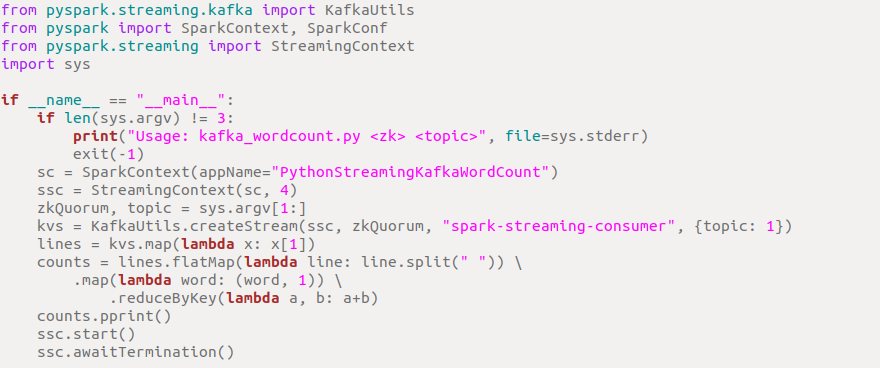


Fig15.SparkStreaming+kafka+flume 实时词频统计源码



Fig16.SparkStreaming+kafka+flume 实时词频统计数据输入

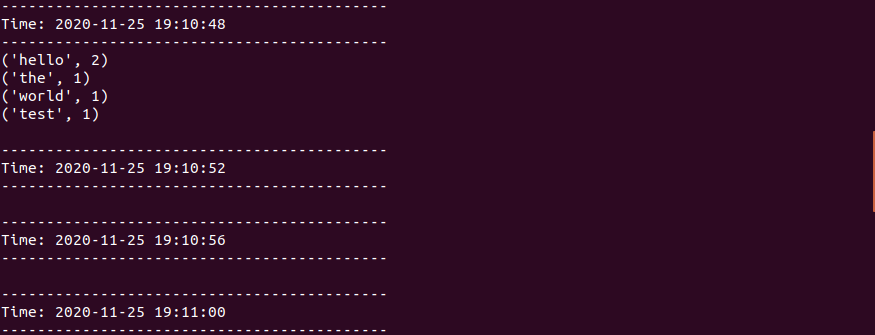


Fig17.SparkStreaming+kafka+flume 实时词频统计统计输出

**3.2模型搭建训练**

**3.2.1数据集的收集**

数据集是在Kaggle上找到的IDS2018，该数据集最初是由纽布伦斯威克大学创建的，用于分析DDoS数据。数据集本身基于大学服务器的日志，该日志在整个公开可用期内发现了各种DoS攻击。在为这些数据编写机器学习笔记本时，“标签”列可以说是数据中最重要的部分，因为它确定发送的数据包是否为恶意数据。

**3.2.2数据集的格式与说明**

我们的数据集是通用的csv文件格式，包含80列，最后一列为标签列。本实验用于训练模型的数据规模达到50万条。

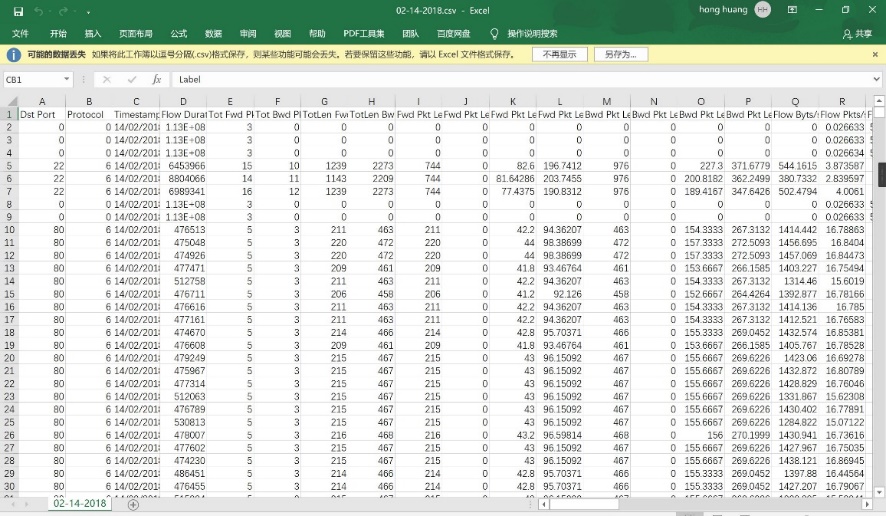


Fig18.数据集格式

**3.2.3数据集的使用**

针对spark的MLlib模块，编写了python脚本将csv文件格式的数据集转换为libsvm格式的txt文件以便对应的API直接使用。

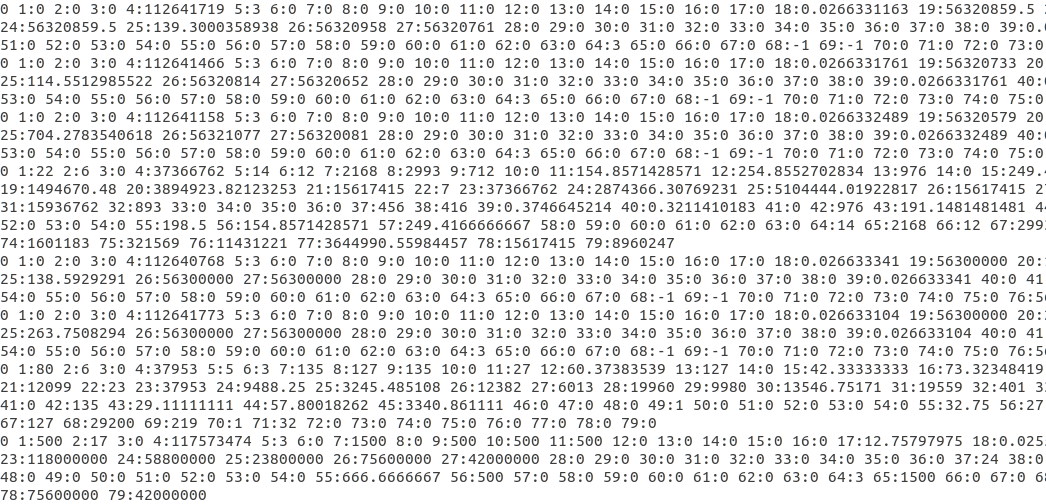


Fig19.libsvm格式数据集

**3.2.4模型设计与对比**

针对本次实验，分别用决策树与神经网络来进行模型训练，取二者准确率较高的一个作为最终模型，以下分别是决策树与神经网络的训练测试结果以及模型结构。

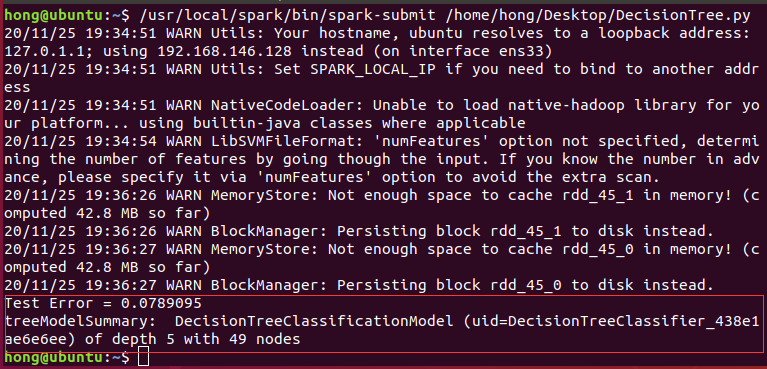


Fig20.决策树训练结果

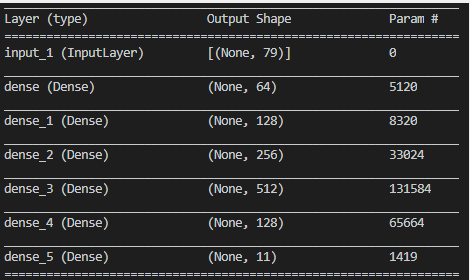




Fig21.神经网络训练结果

通过上述实验结果可以看出决策树的错误率大致为0.078，准确率大致达到0.922，同时运用神经网络训练得到的模型准确率能达到0.96。通过对比最终选择了神经网络训练的模型作为检测器，并把它广播到spark的工作节点，对实时数据进行检测。

**3.3 SparkStreaming实时检测**

本场景下利用SparkStreaming进行实时检测的要点包括如下：

1. 熟悉SparkStreaming基本流程，在本场景下需要对kafka的实时数据进行消费，如图Fig22中118行。
2. 熟悉Dstream的转换操作，包括无状态转换（如map、flatMap、filter、reduceByKey等）、有状态转换（主要的两种类型是滑动窗口和updateStateByKey，前者以一个时间阶段为滑动窗口进行操作，后者则用来跟踪每个键的状态变化。）
3. 需要将相应区间统计结果实时存入redis数据库中，如图Fig25。
4. 需要将训练好的模型，从相应的存储位置取出，并广播到所有工作节点，避免工作节点重复拉取模型，造成性能问题。如图Fig22中114行。
5. 由于没有实际场景的web日志服务器，需要编写相应的脚本模拟日志的产生，将实时产生的日志数据添加到flume监控的日志文件尾端。(在代码中即produce.py脚本)

核心代码如下图：



Fig22.SparkStreaming核心代码

同时可以打印相应的中间结果的输出，更好的理解SparkStreaming的运行流程与原理，在实际运用中熟练运用对DStream的操作。图Fig23展示了对SparkStreaming对DStream的基本操作代码，以及相应中间结果的输出如Fig24(左侧为中间结果打印，右侧为实时产生的日志文件)。

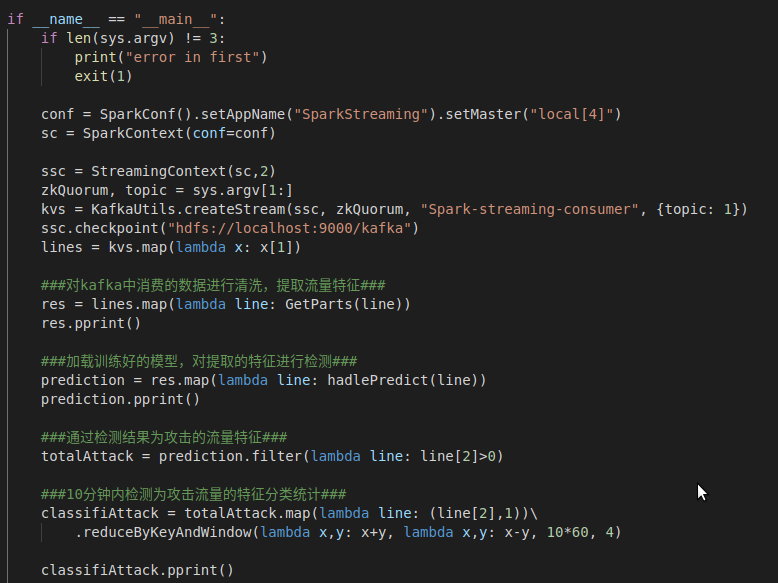


Fig23.SparkStreaming处理过程示例



Fig24.SparkStreaming处理中间结果

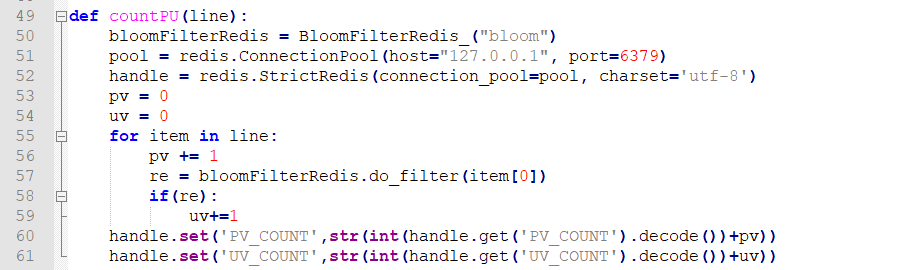


Fig25.SparkStreaming数据存储到redis

**3.3.1数据持久化**

在这之前必须先初始化mysql数据库，建立相应的数据表，核心代码如图Fig26。初始化之后就能通过查询mysql得到图Fig28的结果。

由于实时的检测统计结果临时存放在redis中，需要定时的对结果进行统计存储到MySQL（核心代码如图Fig27所示）,设计了简易的数据库()，对攻击分类（attackedClass），受攻击IP(attackedIP)，受攻击端口(attackedPort)，网站PV\UV(pv\_uv)等数据进行统计，后期对统计结果进行实时展示。

至此所有的准备工作就已经完成，经过一段时间的运行过后，我们再次查看mysql数据库表中的内容，就能看到如图Fig(29-32)所示的新增的数据内容。现在我们就已经完成了本项目最复杂的一部分。

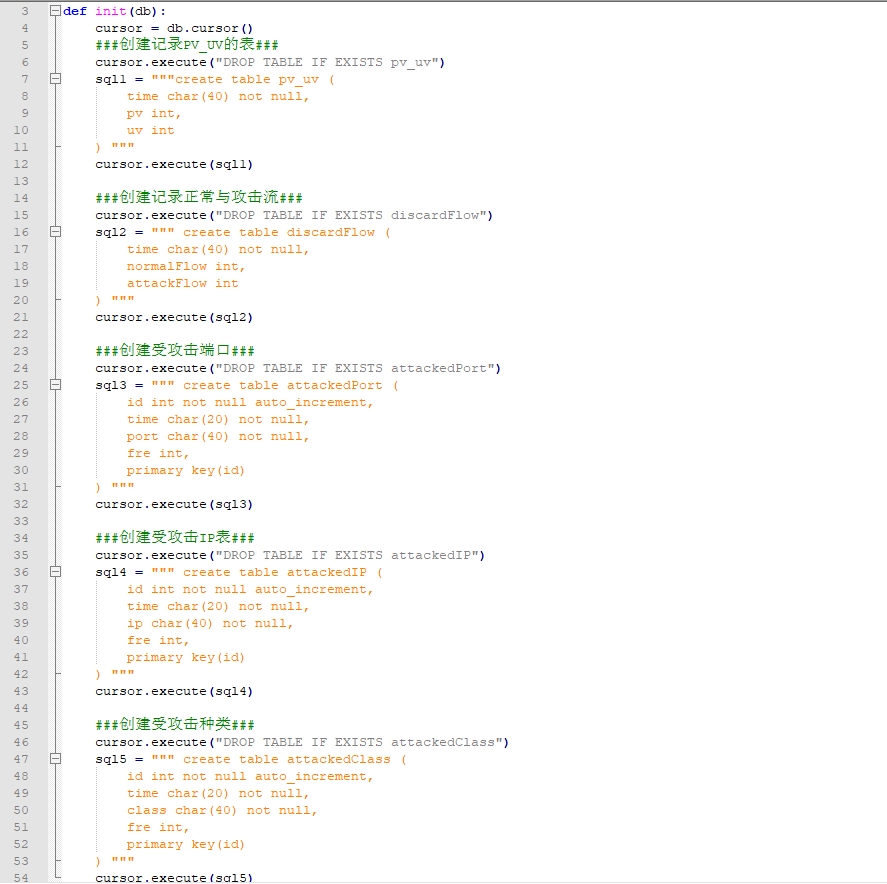


Fig26.mysql初始化



Fig27.获取redis数据持久化到mysql

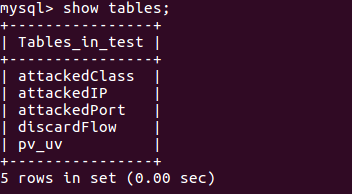


Fig28.mysql初始化表

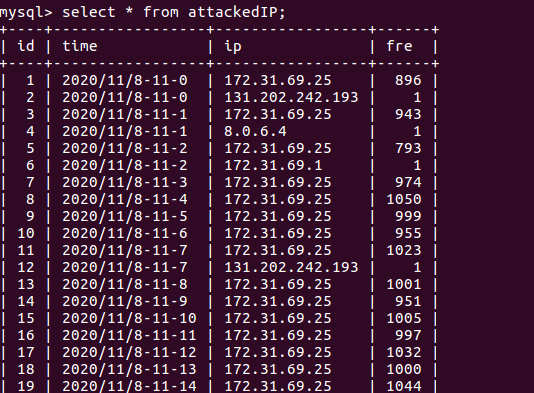
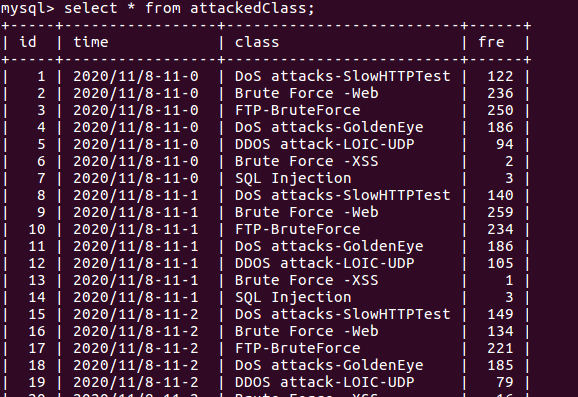
 

Fig29.attackedIP表内容 Fig30.attackedClass表内容

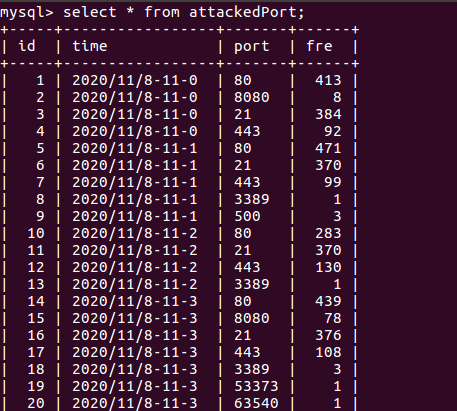
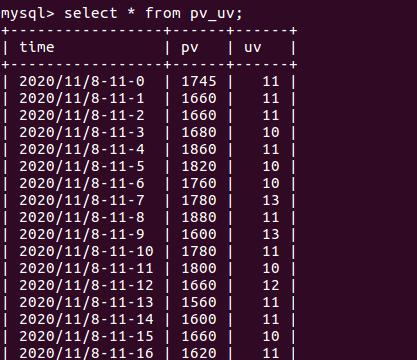
 

Fig31.attackedPort表内容 Fig32.PV/UV表内容

**3.3.2结果可视化**

经过前面的步骤，得到相应的实时生成检测数据在mysql数据库中，为了方便实时的查看与对比，利用python的matplotlib库对检测结果进行一个实时的展示。通过饼状图、折线图、柱状图的方式能很清楚的看到整个网站所受攻击的态势。如Fig33各种网络攻击的总类比例，Fig34遭受攻击的端口信息及频次,Fig35正常访问与非正常访问对比，Fig36网站访问量信息。最后可视化的部分代码如图Fig37所示。

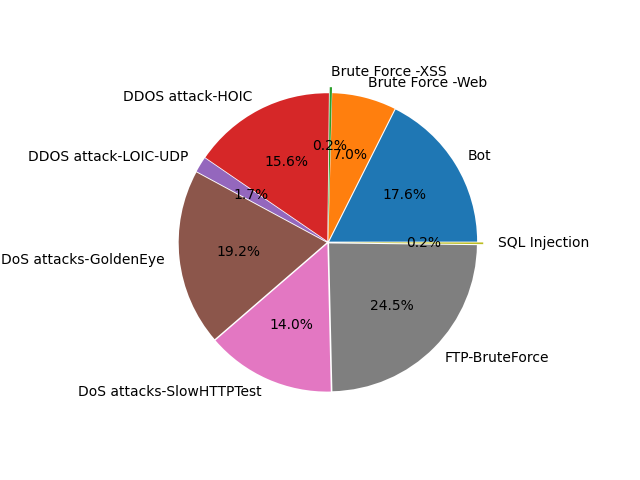


Fig33.各种网络攻击的总类比例

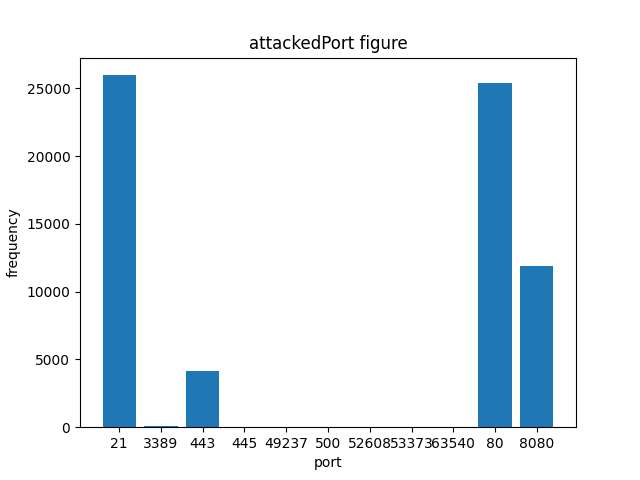


Fig34.遭受攻击的端口信息及频次

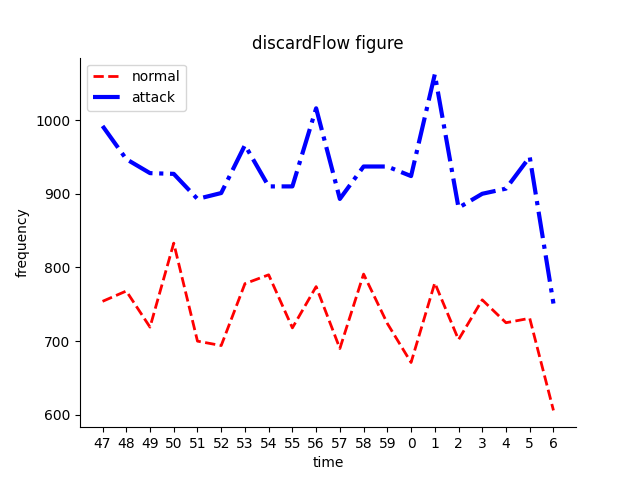


Fig35.正常访问与非正常访问对比

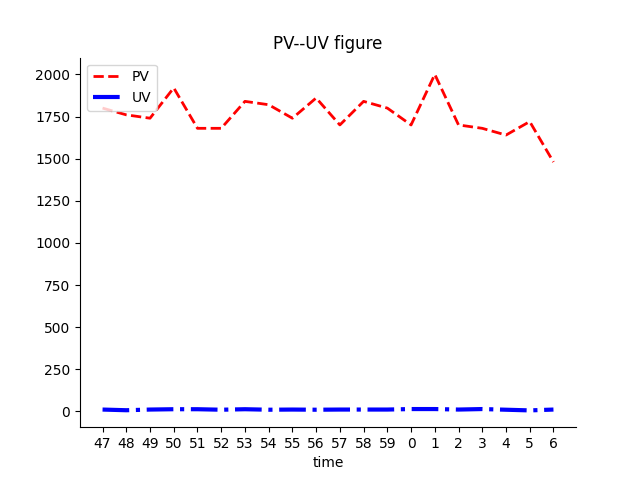


Fig36.网站访问量信息



Fig37.部分可视化代码

## 4 小结

本案例选取基于Spark的实时攻击检测问题作为主题，使用来源于Kaggle平台上的IDS2018入侵检测数据集作为实验数据，结合数据挖掘、机器学习的相关方法进行涡轮风扇发动机的预测性维护研究。主要内容包括为hadoop+spark集群部署、模型搭建训练，以及SparkStreaming实时检测、可视化展示。该案例主题新颖，结合了经济社会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pandas（数据读取与预处理库），scikit-learn（机器学习算法库），matplotlib（可视化绘图库）。

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.

参考内容：数据预处理理论，线性回归、广义线性回归、决策树、SVM、logistic回归算法

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

参考内容：深度神经网络DNN模型