卷　　　号	
卷 内 编 号	
密　　　级	

<项目编号 1.0>

<人机鼠标轨迹识别>

项目开发总结报告

<1.0>

项 目 承 担 部 门：

撰　写　人（签名）：万凤玲

完　　成　　日　　期：2019 年 6 月 22 日

本文档 使 用部门：　☐主管领导　　　☐项目组
☐客户（市场）　☐维护人员　☐用户

评审负责人（签名）：

评　　审　　日　期：

文档信息

标题: 人机鼠标轨迹识别
作者: 万凤玲
创建日期: 2019 年 6 月 23 日
上次更新日期: 2019 年 6 月 22 日
版本: 1.0
部门名称: 万凤玲

修订文档历史记录

日期	版本	说明	作者

目录

1 引言 1

1.1 编写目的 1
1.2 背景 1
1.3 定义 1
1.4 参考资料 2

2 实际开发结果 2

2.1 软件产品描述 2
2.2 主要功能和性能 3
2.3 进度 7

3 开发工作评价 8

3.1 对项目开发过程的评价 8
3.2 对技术方法的评价 8

4 经验与教训 8

5 附录： 9

1 引言

1.1 编写目的

编写这份开发文档的母的是为了提高软件开发的效率，保证软件的质量，而且在软件的使用过程中有指导，帮助，解惑的作用，尤其在维护工作中，文档是不可或缺的资料。

1.2 背景

说明项目的相关背景，包括：

鼠标轨迹识别当前广泛运用于多种人机验证产品中，不仅便于用户的理解记忆，而且极大增加了暴力破解难度。但攻击者可通过黑客工具产生类人轨迹批量操作以绕过检测，并在对抗过程中不断升级其伪造数据以持续绕过同样升级的检测技术。我们期望用机器学习算法来提高人机验证中各种机器行为的检出率，其中包括对抗过程中出现的新的攻击手段的检测；
如今计算机智能识别登录用户是虚拟用户还是实际用户是目前网页登录时采用的一个常用检测手段，以提高网络服务的安全和有效性；
服务端可以采用人工智能手段通过机器学习方式实现准确区分计算机生成的鼠标轨迹和人移动鼠标产生的鼠标轨迹，从而实现拒绝向虚拟用户提供服务而有效利用资源；
软件产品开发者为万凤玲；
软件产品用户主要是一些需要进行登陆验证的网站商家

1.3 定义

Hadoop：Hadoop 是一个由 Apache 基金会所开发的分布式系统基础架构。用户可以在不了解分布式底层细节的情况下，开发分布式程序。充分利用集群的威力进行高速运算和存储。

SPARK：Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎。Spark 是 UC Berkeley AMP lab (加州大学伯克利分校的 AMP 实验室)所开源的类 Hadoop MapReduce 的通用并行框架，Spark，拥有 Hadoop MapReduce 所具有的优点；但不同于 MapReduce 的是——Job 中间输出结果可以保存在内存中，从而不再需要读写 HDFS，因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 MapReduce 的算法。

决策树：决策树是一个预测模型；他代表的是对象属性与对象值之间的一种映射关系。树中每个节点表示某个对象，而每个分叉路径则代表的某个可能的属性值，而每个叶结点则对应从根节点到该叶节点所经历的路径所表示的对象的值。决策树仅有单一输出，若欲有复数输出，可以建立独立的决策树以处理不同输出。数据挖掘中决策树是一种经常要用到的技术，可以用于分析数据，同样也可以用来作预测。

随机森林：随机森林是一个包含多个决策树的分类器， 并且其输出的类别是由个别树输出的类别的众数而定。

1.4 参考资料

列出参考资料文件的标题、文件编号、发表日期和出版单位，说明能够得到这些文件资料的来源。

冯和陈先生。 Learning Apache Spark ，GITHUB 2017。
url:https://github.com/MingChen0919
安东•基里洛夫。Apache Spark：核心概念、架构和内部。
url:http://datastrophic.io/core-concepts-architecture-and-internals-of-apache-spark/
Learning Spark: Lightning-Fast Big Data Analysis (英语) 1st Edition
Holden Karau (Author), Andy Konwinski (Author), Patrick Wendell (Author), Matei Zaharia (Author)
url:https://www.amazon.com/Learning-Spark-Lightning-Fast-Data-Analysis/dp/1449358624
Lean Apache Spark 2 2017-03 Packt Publishing 出版，作者 Muhammad Asif Abbasi

2 实际开发结果

2.1 软件产品描述

说明本次发行、交付的软件产品的版本，包括：

a. 版本标识：1.0
b. 子系统、模块清单：名称、标识、大小：

软件模块清单：

序号	模块名称	模块标识	代码大小
1	清洗数据模块	JavaRDD<String> inf = inp.filter	23 行
2	特征值提取模块	trainingPre testingPre	583 行
3	建模测试模块	RandomForestModel mod = RandomForest.trainClassifier	23 行
代码总量合计	676 行		

c. 文档清单:

文档清单：

序号	文档名称	须提交用户
1	开发文档	万凤玲
2	源代码	万凤玲

c.　同上一次发行版本的区别：

无，本次为 1.0 版本

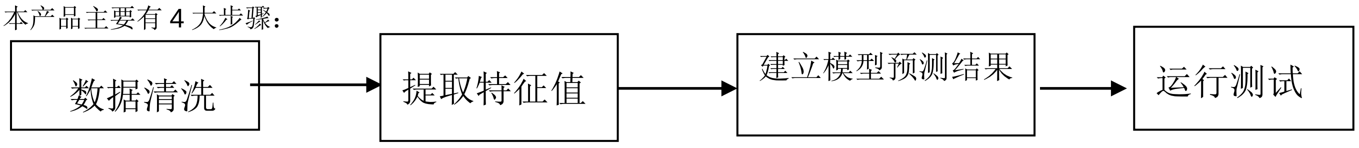
e. 目前的已知缺陷。

性能未能达到 100%

2.2　主要功能和性能

整体框架：

本产品主要有 4 大步骤：



1、　数据清洗：

将训练集与测试集中的数据不合法的数据都过滤掉：

```
SparkConf conf = new SparkConf().setAppName("Mouse").setMaster("local[2]");
JavaSparkContext sc = new JavaSparkContext(conf);
JavaRDD<String> inp  = sc.textFile(inputPath);
JavaRDD<String> tnp  = sc.textFile(inputPath2);
JavaRDD<String> inf = inp.filter(f -> {
    String[] arr = f.split(" ");
    if(arr.length != 4)//训练集字段小于4为无效数据
        return false;
    else if((arr[1].split(";")).length < 2)//只有一个点的轨迹过滤掉
        return false;
    else
        return true;
});
JavaRDD<String> tnf = tnp.filter(f -> {
    String[] arr = f.split(" ");
    if(arr.length != 3)//测试集小于3个字段的为无效数据
        return false;
    else if((arr[1].split(";")).length < 2)//只有一个点的轨迹过滤掉
        return false;
    else
        return true;
});
```

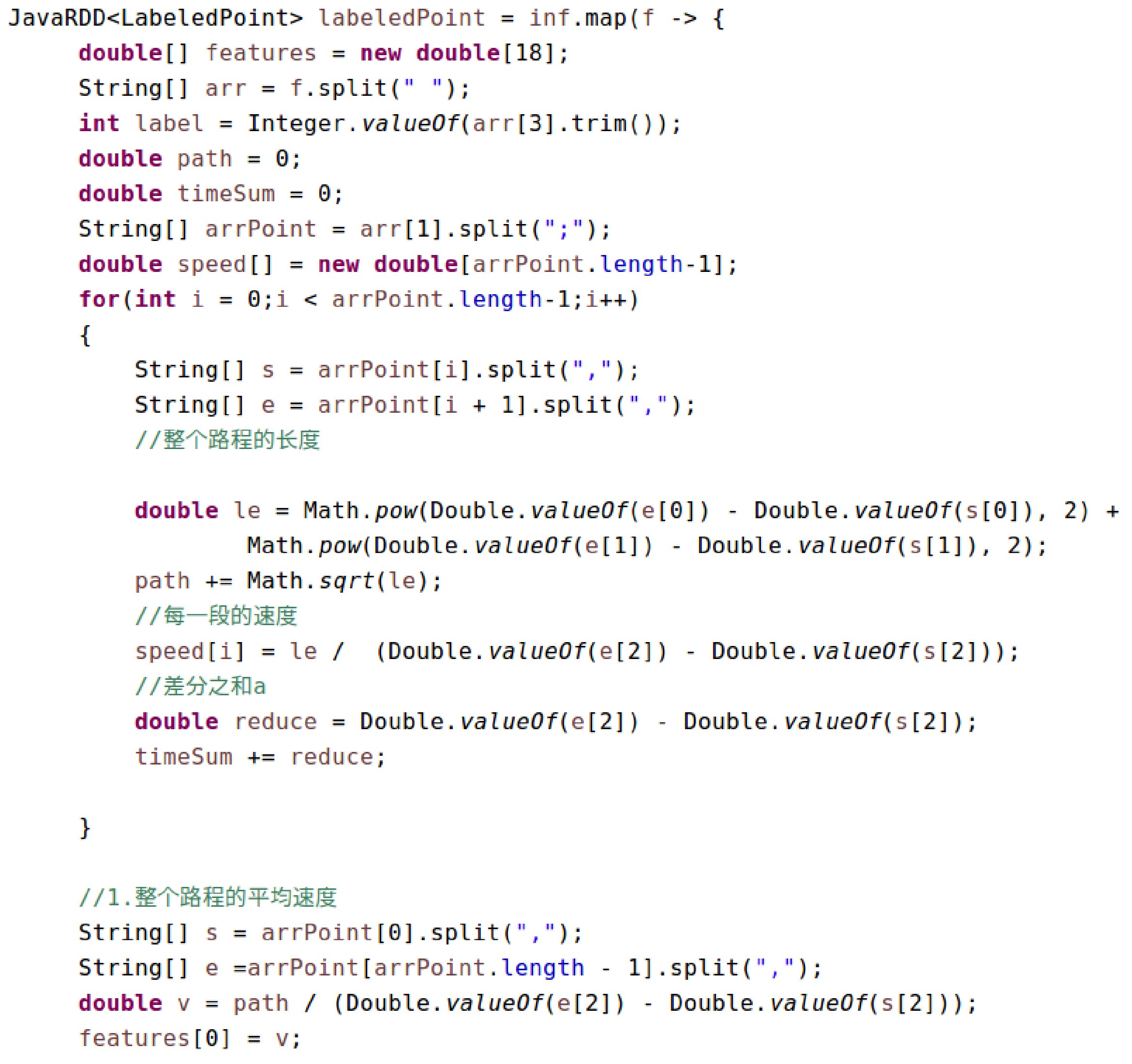
2、　提取特征值

一共提取了 18 种特征值（训练集与特征值类似）：

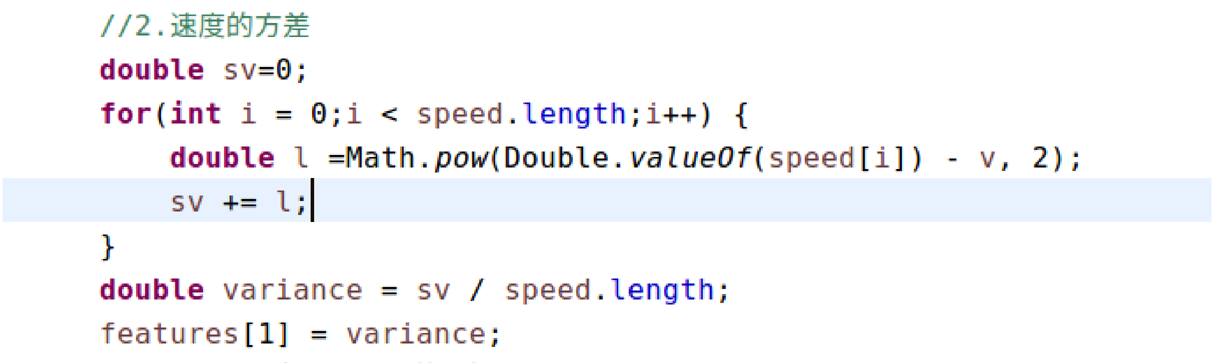
整个路程的平均速度、速度的方差、速度的一阶差分平均值、最大速度、平均加速度、最大加速度、轨迹 x 方向走一半花的时间比、走一半路花的时间占总时间比、停顿时间占总时间、x 坐标回退次数占总次数 、时间轴 t 差分后的均值 、t 进行一阶差分后的标准差 、平均角度偏移、平均角度偏移标

准差、同一时间坐标是否变化、最大角度、相邻点构成的角度序列一阶差分后的均值

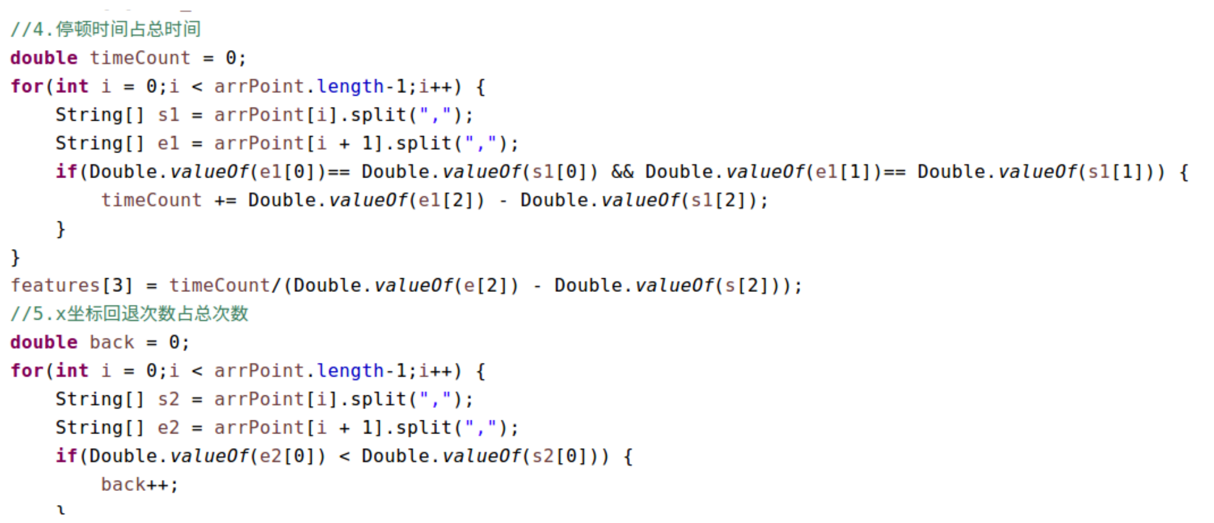
主要特征值提取代码：



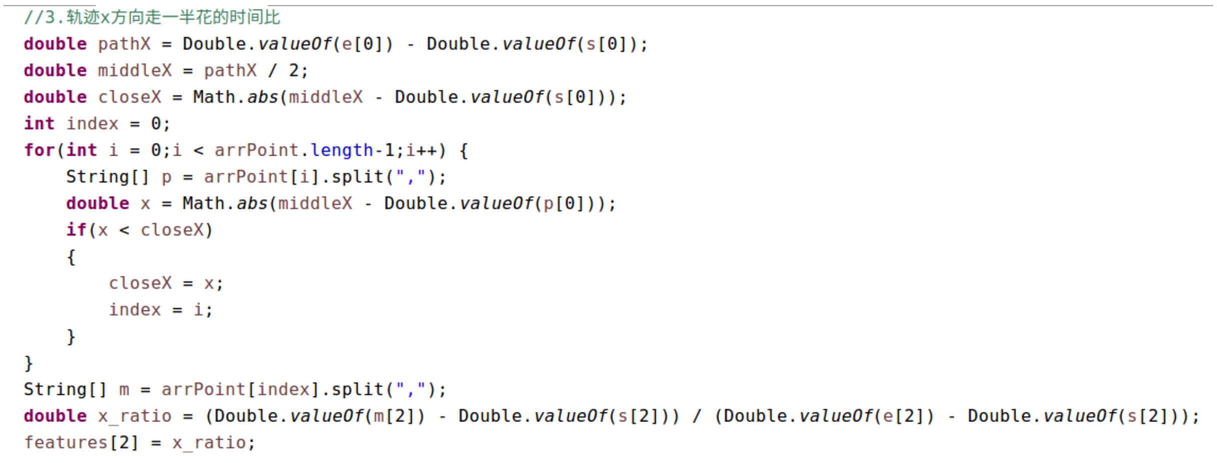
```java
JavaRDD<LabeledPoint> labeledPoint = inf.map(f -> {
    double[] features = new double[18];
    String[] arr = f.split(" ");
    int label = Integer.valueOf(arr[3].trim());
    double path = 0;
    double timeSum = 0;
    String[] arrPoint = arr[1].split(";");
    double speed[] = new double[arrPoint.length-1];
    for(int i = 0;i < arrPoint.length-1;i++)
    {
        String[] s = arrPoint[i].split(",");
        String[] e = arrPoint[i + 1].split(",");
        //整个路程的长度

        double le = Math.pow(Double.valueOf(e[0]) - Double.valueOf(s[0]), 2) +
                Math.pow(Double.valueOf(e[1]) - Double.valueOf(s[1]), 2);
        path += Math.sqrt(le);
        //每一段的速度
        speed[i] = le /  (Double.valueOf(e[2]) - Double.valueOf(s[2]));
        //差分之和a
        double reduce = Double.valueOf(e[2]) - Double.valueOf(s[2]);
        timeSum += reduce;

    }

    //1.整个路程的平均速度
    String[] s = arrPoint[0].split(",");
    String[] e =arrPoint[arrPoint.length - 1].split(",");
    double v = path / (Double.valueOf(e[2]) - Double.valueOf(s[2]));
    features[0] = v;
```



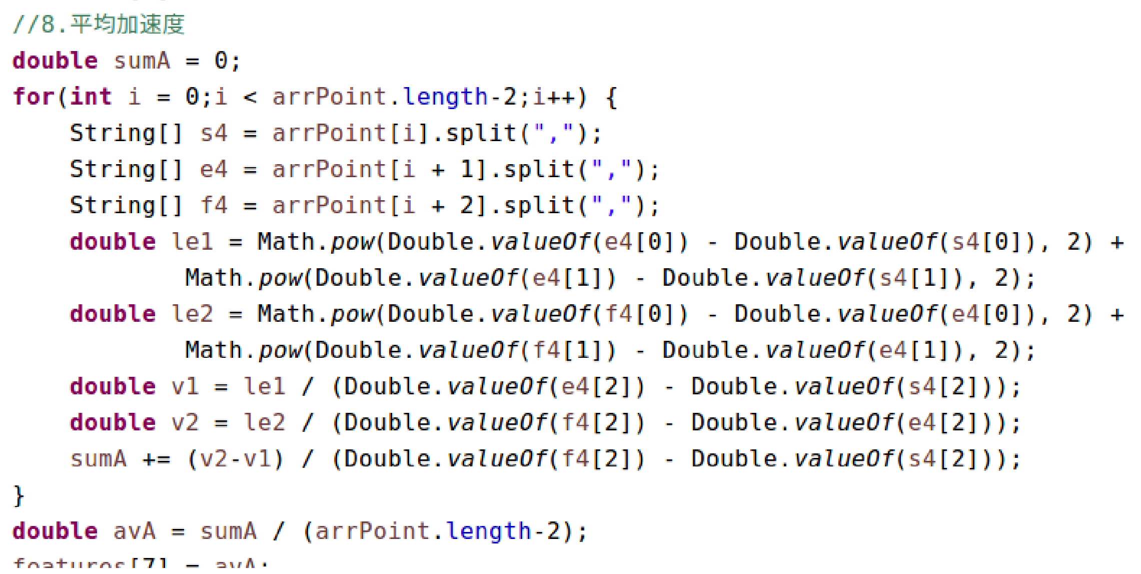
```java
    //2.速度的方差
    double sv=0;
    for(int i = 0;i < speed.length;i++) {
        double l =Math.pow(Double.valueOf(speed[i]) - v, 2);
        sv += l;
    }
    double variance = sv / speed.length;
    features[1] = variance;
```



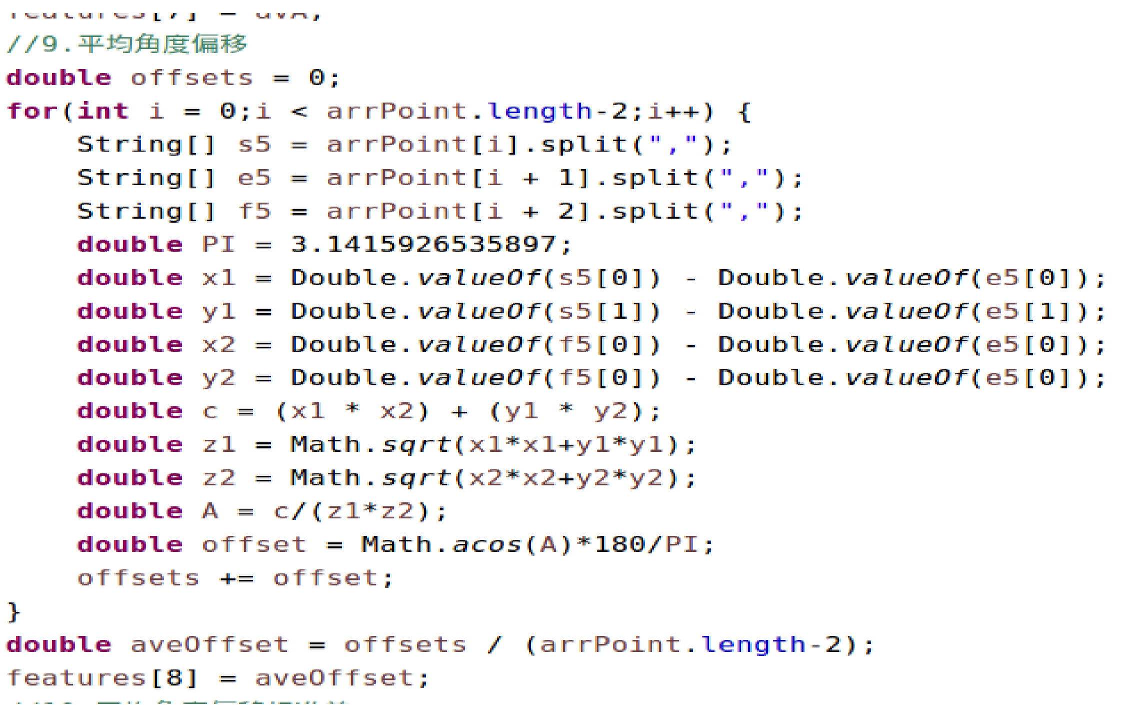
```java
//4.停顿时间占总时间
double timeCount = 0;
for(int i = 0;i < arrPoint.length-1;i++) {
    String[] s1 = arrPoint[i].split(",");
    String[] e1 = arrPoint[i + 1].split(",");
    if(Double.valueOf(e1[0])== Double.valueOf(s1[0]) && Double.valueOf(e1[1])== Double.valueOf(s1[1])) {
        timeCount += Double.valueOf(e1[2]) - Double.valueOf(s1[2]);
    }
}
features[3] = timeCount/(Double.valueOf(e[2]) - Double.valueOf(s[2]));
//5.x坐标回退次数占总次数
double back = 0;
for(int i = 0;i < arrPoint.length-1;i++) {
    String[] s2 = arrPoint[i].split(",");
    String[] e2 = arrPoint[i + 1].split(",");
    if(Double.valueOf(e2[0]) < Double.valueOf(s2[0])) {
        back++;
    }
```



```java
//3.轨迹x方向走一半花的时间比
double pathX = Double.valueOf(e[0]) - Double.valueOf(s[0]);
double middleX = pathX / 2;
double closeX = Math.abs(middleX - Double.valueOf(s[0]));
int index = 0;
for(int i = 0;i < arrPoint.length-1;i++) {
    String[] p = arrPoint[i].split(",");
    double x = Math.abs(middleX - Double.valueOf(p[0]));
    if(x < closeX)
    {
        closeX = x;
        index = i;
    }
}
String[] m = arrPoint[index].split(",");
double x_ratio = (Double.valueOf(m[2]) - Double.valueOf(s[2])) / (Double.valueOf(e[2]) - Double.valueOf(s[2]));
features[2] = x_ratio;
```



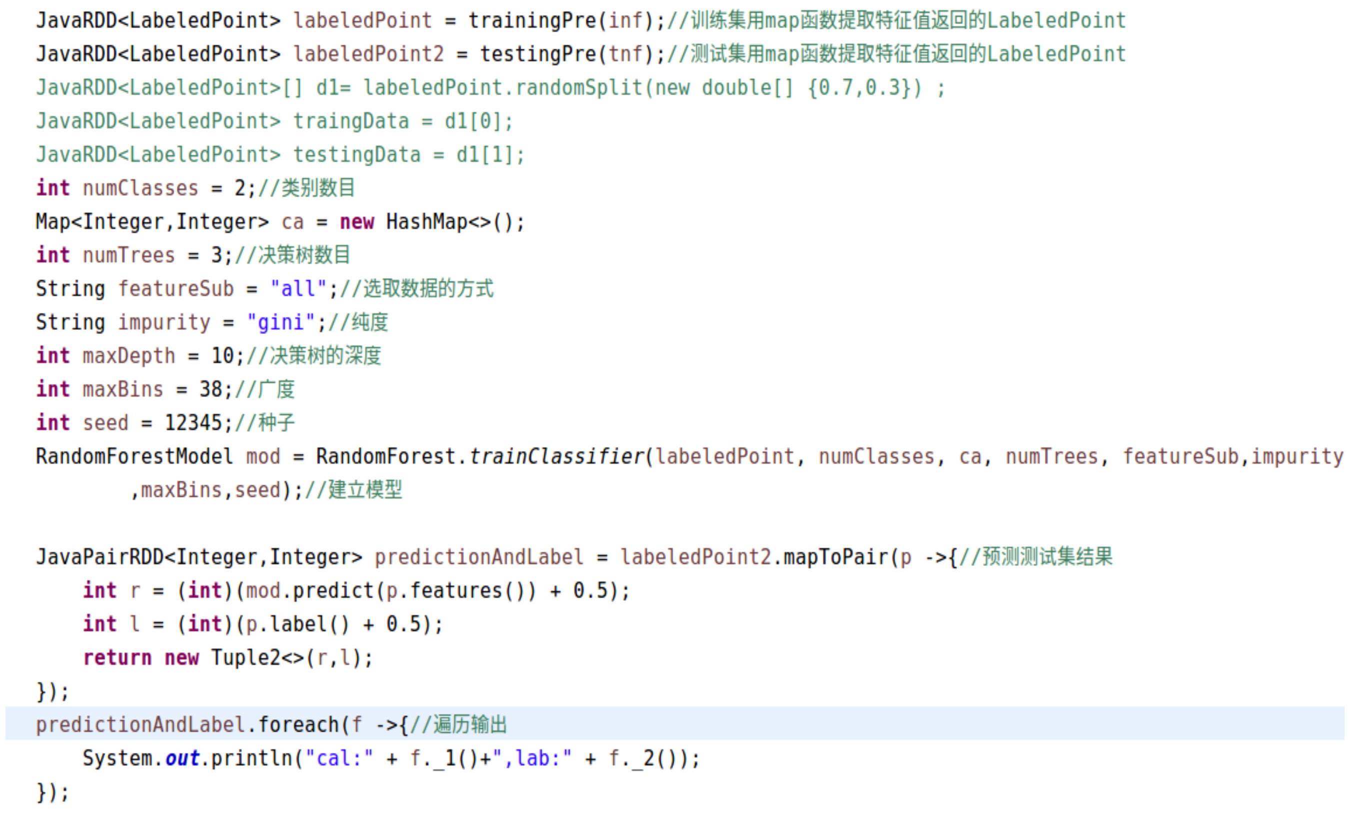
```java
//8.平均加速度
double sumA = 0;
for(int i = 0;i < arrPoint.length-2;i++) {
    String[] s4 = arrPoint[i].split(",");
    String[] e4 = arrPoint[i + 1].split(",");
    String[] f4 = arrPoint[i + 2].split(",");
    double le1 = Math.pow(Double.valueOf(e4[0]) - Double.valueOf(s4[0]), 2) +
            Math.pow(Double.valueOf(e4[1]) - Double.valueOf(s4[1]), 2);
    double le2 = Math.pow(Double.valueOf(f4[0]) - Double.valueOf(e4[0]), 2) +
            Math.pow(Double.valueOf(f4[1]) - Double.valueOf(e4[1]), 2);
    double v1 = le1 / (Double.valueOf(e4[2]) - Double.valueOf(s4[2]));
    double v2 = le2 / (Double.valueOf(f4[2]) - Double.valueOf(e4[2]));
    sumA += (v2-v1) / (Double.valueOf(f4[2]) - Double.valueOf(s4[2]));
}
double avA = sumA / (arrPoint.length-2);
features[7] = avA;
```



```
features[7] = avA;
//9.平均角度偏移
double offsets = 0;
for(int i = 0;i < arrPoint.length-2;i++) {
    String[] s5 = arrPoint[i].split(",");
    String[] e5 = arrPoint[i + 1].split(",");
    String[] f5 = arrPoint[i + 2].split(",");
    double PI = 3.1415926535897;
    double x1 = Double.valueOf(s5[0]) - Double.valueOf(e5[0]);
    double y1 = Double.valueOf(s5[1]) - Double.valueOf(e5[1]);
    double x2 = Double.valueOf(f5[0]) - Double.valueOf(e5[0]);
    double y2 = Double.valueOf(f5[0]) - Double.valueOf(e5[0]);
    double c = (x1 * x2) + (y1 * y2);
    double z1 = Math.sqrt(x1*x1+y1*y1);
    double z2 = Math.sqrt(x2*x2+y2*y2);
    double A = c/(z1*z2);
    double offset = Math.acos(A)*180/PI;
    offsets += offset;
}
double aveOffset = offsets / (arrPoint.length-2);
features[8] = aveOffset;
```

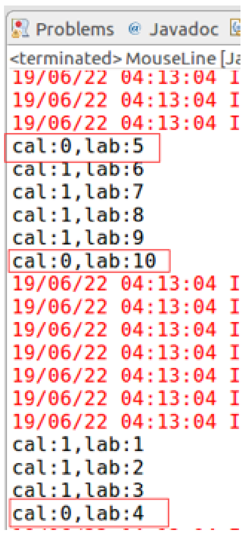
3、 建立模型预测结果

建立随机森林模型，预测测试集中的数据是人为还是机器：

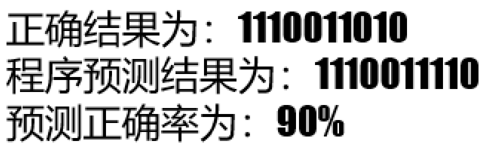


```
JavaRDD<LabeledPoint> labeledPoint = trainingPre(inf);//训练集用map函数提取特征值返回的LabeledPoint
JavaRDD<LabeledPoint> labeledPoint2 = testingPre(tnf);//测试集用map函数提取特征值返回的LabeledPoint
JavaRDD<LabeledPoint>[] d1= labeledPoint.randomSplit(new double[] {0.7,0.3}) ;
JavaRDD<LabeledPoint> traingData = d1[0];
JavaRDD<LabeledPoint> testingData = d1[1];
int numClasses = 2;//类别数目
Map<Integer,Integer> ca = new HashMap<>();
int numTrees = 3;//决策树数目
String featureSub = "all";//选取数据的方式
String impurity = "gini";//纯度
int maxDepth = 10;//决策树的深度
int maxBins = 38;//广度
int seed = 12345;//种子
RandomForestModel mod = RandomForest.trainClassifier(labeledPoint, numClasses, ca, numTrees, featureSub,impurity
        ,maxBins,seed);//建立模型

JavaPairRDD<Integer,Integer> predictionAndLabel = labeledPoint2.mapToPair(p ->{//预测测试集结果
    int r = (int)(mod.predict(p.features()) + 0.5);
    int l = (int)(p.label() + 0.5);
    return new Tuple2<>(r,l);
});
predictionAndLabel.foreach(f ->{//遍历输出
    System.out.println("cal:" + f._1()+",lab:" + f._2());
});
```

4、 运行测试



```
<terminated> MouseLine [Ja
19/06/22 04:13:04 I
19/06/22 04:13:04 I
19/06/22 04:13:04 I
cal:0,lab:5
cal:1,lab:6
cal:1,lab:7
cal:1,lab:8
cal:1,lab:9
cal:0,lab:10
19/06/22 04:13:04 I
19/06/22 04:13:04 I
19/06/22 04:13:04 I
19/06/22 04:13:04 I
19/06/22 04:13:04 I
19/06/22 04:13:04 I
19/06/22 04:13:04 I
cal:1,lab:1
cal:1,lab:2
cal:1,lab:3
cal:0,lab:4
```



正确结果为：1110011010
程序预测结果为：1110011110
预测正确率为：90%

主要函数：

主函数 main

拥有数据清洗、调用提取征值函数、建模测试的函数 mousePre

训练集提取特征值函数 trainingPre

测试集提取特征值函数 testingPre

实现过程描述：

1、用 JavaSparkContext 的 textFile 函数提取文件中的数据转为 JavaRDD<String>

2、用 filter 函数过滤数据

3、在 map 函数中提取特征值，返回 JavaRDD<LabeledPoint>型的值

4、用随机森林模型建模并测试得到结果

性能分析：

通过最后测试，有 90%的正确率。说明正确率较高，性能较好，在很多场合都能够适用。

2.3 进度

列出原定计划进度与实际进度的对比，明确说明，实际进度是提前了、还是延迟了，分析主要原因。可使用如下表头：

阶段	计划工作日	实际工作日	对比	原因
数据分析阶段	1 天	0.5 天	提前 0.5 天	效率高效
数据清洗阶段	0.5 天	0.5 天	按时完成	步骤简单
特征值提取阶段	1 天	1． 5 天	延时 0.5 天完成	步骤繁琐
建模测试阶段	1 天	1 天	按时完成	效率高效
模型调优测试阶段	0.5 天	0.5 天	按时完成	效率高效

3 开发工作评价

3.1 对项目开发过程的评价

数据分析阶段：在指导老师的指导下，数据分析比较顺利，完成情况良好。训练集有 4 个数据段，测试集有 3 个数据段；

数据清洗阶段：对数据的清洗完成度较好，不仅完成了对字段数目不够的数据的清洗，也完成了对无效轨迹的过滤；

提取特征值阶段：本阶段耗时很长，是本项目中最为耗时与重要的阶段。我一共提取了 18 种特征值，这些特征值的提取需要不同的逻辑与不同的数学公式才能完成，所以比较繁琐。完成情况比较良好，一些特征值因为所给数据的原因，所以精确度不是很高，但是也基本能够识别轨迹的类别；

建模测试阶段：在本阶段，使用了随机森林模型进行建模。使用训练集中的数据特征值进行训练，再使用 predict 函数进行预测，预测测试集中的轨迹的类别。本阶段步骤比较简单，但是建模的参数需要进行调整（如决策树数目等），参数调优比较繁琐；

开发文档编写阶段：编写开发文档时，只需根据自己的思路与开发程序的思路进行编写。编写过程比较顺利，未遇到什么大的问题；

遇到的一些问题：在测试阶段，不仅需要调整模型的参数，同时也要注意特征值的添加。

3.2 对技术方法的评价

Spark：Spark 很快，支持交互式计算和复杂算法。

决策树：速度快，计算量相对较小，且容易转化成分类规则. 只要沿着树根向下一直走到叶，沿途的分裂条件就能够唯一确定一条分类的谓词.准确性高，挖掘出来的分类规则准确性高，便于理解，决策树可以清晰的显示哪些字段比较重要，即可以生成可以理解的规则.可以处理连续和种类字段不需要任何领域知识和参数假设，适合高维数据。

随机森林：在当前的很多数据集上，相对其他算法有着很大的优势，表现良好；它能够处理很高维度的数据，并且不用做特征选择，因为特征子集是随机选择的；在训练完后，它能够得出特征重要性；随机森林有 oob，不需要单独换分交叉验证集；训练时树与树之间是相互独立的，训练速度快，容易做成并行化方法；对缺失值不敏感，如果有很大一部分的特征遗失，仍可以维持准确度。

4 经验与教训

在这次项目开发中，我得到了许多经验与教训。在分析数据时，一定要仔细，明白数据代表的含义，因为数据分析是项目一个很重要的阶段。如果数据一旦分析错误，那么你后面的编码方向就完全错误。在提取特征值时，并不是特征值越多越好，而是要提取关键有用的特征值。在测试阶段，一定要有耐心，要变化各种参数得到最优的模型。

总的来说，这次项目让我对决策树和随机森林模型有了更深的了解，因为运用了才能使印象更加深刻。

5 附录：

```
package net.mouse.line;



import java.util.HashMap;
import java.util.Map;

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.mllib.regression.LabeledPoint;
import org.apache.spark.mllib.tree.RandomForest;
import org.apache.spark.mllib.tree.model.RandomForestModel;

import scala.Tuple2;

import org.apache.spark.mllib.linalg.Vectors;
public class MouseLine {
	private final String inputPath = "/home/wfl/Documents/dsjtzs_txfz_training.txt";
	private final String inputPath2 = "/home/wfl/Documents/dsjtzs_txfz_test_combine.txt";
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		try {
				MouseLine m = new MouseLine();
				m.mousePre();
			}catch(Exception e) {
				e.printStackTrace();
			}
	}



	private void mousePre() throws Exception
	{
		SparkConf conf = new SparkConf().setAppName("Mouse").setMaster("local[2]");
		 JavaSparkContext sc = new JavaSparkContext(conf);
		 JavaRDD<String> inp    = sc.textFile(inputPath);
		 JavaRDD<String> tnp    = sc.textFile(inputPath2);
		 JavaRDD<String> inf = inp.filter(f -> {
```

```
            String[] arr = f.split(" ");
            if(arr.length != 4)//训练集字段小于4为无效数据
                return false;
            else if((arr[1].split(";")).length < 2)//只有一个点的轨迹过滤掉
                return false;
            else
                return true;
        });
        JavaRDD<String> tnf = tnp.filter(f -> {
            String[] arr = f.split(" ");
            if(arr.length != 3)//测试集小于3个字段的为无效数据
                return false;
            else if((arr[1].split(";")).length < 2)//只有一个点的轨迹过滤掉
                return false;
            else
                return true;
        });
        JavaRDD<LabeledPoint> labeledPoint = trainingPre(inf);//训练集用map函数提取
特征值返回的LabeledPoint
        JavaRDD<LabeledPoint> labeledPoint2 = testingPre(tnf);//测试集用map函数提取
特征值返回的LabeledPoint
//      JavaRDD<LabeledPoint>[] d1= labeledPoint.randomSplit(new double[] {0.7,0.3}) ;
//      JavaRDD<LabeledPoint> traingData = d1[0];
//      JavaRDD<LabeledPoint> testingData = d1[1];
        int numClasses = 2;//类别数目
        Map<Integer,Integer> ca = new HashMap<>();
        int numTrees = 3;//决策树数目
        String featureSub = "all";//选取数据的方式
        String impurity = "gini";//纯度
        int maxDepth = 10;//决策树的深度
        int maxBins = 38;//广度
        int seed = 12345;//种子
        RandomForestModel mod = RandomForest.trainClassifier(labeledPoint,
numClasses, ca, numTrees, featureSub,impurity,maxDepth
                ,maxBins,seed);//建立模型

        JavaPairRDD<Integer,Integer> predictionAndLabel = labeledPoint2.mapToPair(p
->{//预测测试集结果
            int r = (int)(mod.predict(p.features()) + 0.5);
            int l = (int)(p.label() + 0.5);
            return new Tuple2<>(r,l);
        });
```

```
            predictionAndLabel.foreach(f ->{//遍历输出
                System.out.println("cal:" + f._1()+",lab:" + f._2());
            });
//              double testErr = predictionAndLabel.filter(pl ->{
//                  if(pl._1() == pl._2())
//                      return false;
//                  else
//                      return true;
//              }).count() /(1.0* labeledPoint.count());
//              System.out.println("Error rating:" + testErr);


        }
        private JavaRDD<LabeledPoint> trainingPre(JavaRDD<String> inf) throws Exception{
            //提取特征值
             JavaRDD<LabeledPoint> labeledPoint = inf.map(f -> {
                double[] features = new double[18];
                String[] arr = f.split(" ");
                int label = Integer.valueOf(arr[3].trim());
                double path = 0;
                double timeSum = 0;
                String[] arrPoint = arr[1].split(";");
                double speed[] = new double[arrPoint.length-1];
                for(int i = 0;i < arrPoint.length-1;i++)
                {
                    String[] s = arrPoint[i].split(",");
                    String[] e = arrPoint[i + 1].split(",");
                    //整个路程的长度

                    double le = Math.pow(Double.valueOf(e[0]) - Double.valueOf(s[0]), 2) +
                            Math.pow(Double.valueOf(e[1]) - Double.valueOf(s[1]), 2);
                    path += Math.sqrt(le);
                    //每一段的速度
                    speed[i] = le /   (Double.valueOf(e[2]) - Double.valueOf(s[2]));
                    //差分之和a
                    double reduce = Double.valueOf(e[2]) - Double.valueOf(s[2]);
                    timeSum += reduce;

                }

                //1.整个路程的平均速度
                String[] s = arrPoint[0].split(",");
```

```
		String[] e =arrPoint[arrPoint.length - 1].split(",");
		double v = path / (Double.valueOf(e[2]) - Double.valueOf(s[2]));
		features[0] = v;
		//2.速度的方差
		double sv=0;
		for(int i = 0;i < speed.length;i++) {
			double l =Math.pow(Double.valueOf(speed[i]) - v, 2);
			sv += l;
		}
		double variance = sv / speed.length;
		features[1] = variance;
		//3.轨迹x方向走一半花的时间比
		double pathX = Double.valueOf(e[0]) - Double.valueOf(s[0]);
		double middleX = pathX / 2;
		double closeX = Math.abs(middleX - Double.valueOf(s[0]));
		int index = 0;
		for(int i = 0;i < arrPoint.length-1;i++) {
			String[] p = arrPoint[i].split(",");
			double x = Math.abs(middleX - Double.valueOf(p[0]));
			if(x < closeX)
			{
				closeX = x;
				index = i;
			}
		}
		String[] m = arrPoint[index].split(",");
		double x_ratio = (Double.valueOf(m[2]) - Double.valueOf(s[2])) /
(Double.valueOf(e[2]) - Double.valueOf(s[2]));
		features[2] = x_ratio;
		//4.停顿时间占总时间
		double timeCount = 0;
		for(int i = 0;i < arrPoint.length-1;i++) {
			String[] s1 = arrPoint[i].split(",");
			String[] e1 = arrPoint[i + 1].split(",");
			if(Double.valueOf(e1[0])== Double.valueOf(s1[0]) &&
Double.valueOf(e1[1])== Double.valueOf(s1[1])) {
				timeCount += Double.valueOf(e1[2]) - Double.valueOf(s1[2]);
			}
		}
		features[3] = timeCount/(Double.valueOf(e[2]) - Double.valueOf(s[2]));
		//5.x坐标回退次数占总次数
		double back = 0;
```

```
for(int i = 0;i < arrPoint.length-1;i++) {
	String[] s2 = arrPoint[i].split(",");
	String[] e2 = arrPoint[i + 1].split(",");
	if(Double.valueOf(e2[0]) < Double.valueOf(s2[0])) {
		back++;
	}
}
features[4] = back/(arrPoint.length-1);
//6.时间轴t差分后的均值
double ave = timeSum/(arrPoint.length-2);
features[5] = ave;
//7.t进行一阶差分后的标准差
double sumRe = 0;
for(int i = 0;i < arrPoint.length-2;i++) {
	String[] s3 = arrPoint[i].split(",");
	String[] e3 = arrPoint[i + 1].split(",");
	double reduce = Double.valueOf(e3[2]) - Double.valueOf(s3[2]);
	double reducePow = Math.pow(reduce - ave, 2);
	sumRe += reducePow;
}
double stan = Math.sqrt(sumRe / (arrPoint.length-2));
features[6] = stan;
//8.平均加速度
double sumA = 0;
for(int i = 0;i < arrPoint.length-2;i++) {
	String[] s4 = arrPoint[i].split(",");
	String[] e4 = arrPoint[i + 1].split(",");
	String[] f4 = arrPoint[i + 2].split(",");
	double le1 = Math.pow(Double.valueOf(e4[0]) - Double.valueOf(s4[0]), 2)
+
			Math.pow(Double.valueOf(e4[1]) - Double.valueOf(s4[1]), 2);
	double le2 = Math.pow(Double.valueOf(f4[0]) - Double.valueOf(e4[0]), 2)
+
			Math.pow(Double.valueOf(f4[1]) - Double.valueOf(e4[1]), 2);
	double v1 = le1 / (Double.valueOf(e4[2]) - Double.valueOf(s4[2]));
	double v2 = le2 / (Double.valueOf(f4[2]) - Double.valueOf(e4[2]));
	sumA += (v2-v1) / (Double.valueOf(f4[2]) - Double.valueOf(s4[2]));
}
double avA = sumA / (arrPoint.length-2);
features[7] = avA;
//9.平均角度偏移
double offsets = 0;
```

```
for(int i = 0;i < arrPoint.length-2;i++) {
	String[] s5 = arrPoint[i].split(",");
	String[] e5 = arrPoint[i + 1].split(",");
	String[] f5 = arrPoint[i + 2].split(",");
	double PI = 3.1415926535897;
	double x1 = Double.valueOf(s5[0]) - Double.valueOf(e5[0]);
	double y1 = Double.valueOf(s5[1]) - Double.valueOf(e5[1]);
	double x2 = Double.valueOf(f5[0]) - Double.valueOf(e5[0]);
	double y2 = Double.valueOf(f5[0]) - Double.valueOf(e5[0]);
	double c = (x1 * x2) + (y1 * y2);
	double z1 = Math.sqrt(x1*x1+y1*y1);
	double z2 = Math.sqrt(x2*x2+y2*y2);
	double A = c/(z1*z2);
	double offset = Math.acos(A)*180/PI;
	offsets += offset;
}
double aveOffset = offsets / (arrPoint.length-2);
features[8] = aveOffset;
//10.平均角度偏移标准差
double sumOff = 0;
for(int i = 0;i < arrPoint.length-2;i++) {
	String[] s6 = arrPoint[i].split(",");
	String[] e6 = arrPoint[i + 1].split(",");
	String[] f6 = arrPoint[i + 2].split(",");
	double PI_A = 3.1415926535897;
	double x3 = Double.valueOf(s6[0]) - Double.valueOf(e6[0]);
	double y3 = Double.valueOf(s6[1]) - Double.valueOf(e6[1]);
	double x4 = Double.valueOf(f6[0]) - Double.valueOf(e6[0]);
	double y4 = Double.valueOf(f6[0]) - Double.valueOf(e6[0]);
	double c1 = (x3 * x4) + (y3 * y4);
	double z3 = Math.sqrt(x3*x3+y3*y3);
	double z4 = Math.sqrt(x4*x4+y4*y4);
	double A1 = c1/(z3*z4);
	double offset = Math.acos(A1)*180/PI_A;
	double subPow = Math.pow(offset - aveOffset, 2);
	sumOff += subPow;
}
double offStand = Math.sqrt(sumOff / (arrPoint.length-2));
features[9] = offStand;
//11.速度的一阶差分平均值
double sumSub = 0;
for(int i = 0;i < speed.length-1;i++)
```

```
		{
			double sub = speed[i+1] - speed[i];
			sumSub += sub;
		}
		double aveSub = sumSub / (speed.length-1);
		features[10] = aveSub;
		//12.同一时间坐标是否变化
		double isStop = 0;
		for(int i = 0;i < arrPoint.length-1;i++) {
			String[] s7 = arrPoint[i].split(",");
			String[] e7 = arrPoint[i + 1].split(",");
	
	if((Double.valueOf(s7[0])!=Double.valueOf(e7[0])||Double.valueOf(s7[1])!=Double.valu
eOf(e7[1]))
					&&Double.valueOf(s7[2])==Double.valueOf(e7[2])) {
				isStop = 1;
			}
		}
		features[11] = isStop;
		//13.走一半路花的时间占总时间比
		double half_path = path/2;
		double path2 = 0;
		double min_diff_path = 100000;
		int index2 = 0;
		for(int i=0;i< arrPoint.length-1;i++)
		{
			String[] s8 = arrPoint[i].split(",");
			String[] e8 = arrPoint[i + 1].split(",");
			
			
			double l = Math.pow(Double.valueOf(e8[0]) - Double.valueOf(s8[0]), 2) +
					Math.pow(Double.valueOf(e8[1]) - Double.valueOf(s8[1]), 2);
			path2 += Math.sqrt(l);
			if(Math.abs(half_path - path2) < min_diff_path) {
				min_diff_path = Math.abs(half_path-path2);
				index = i;
			}
			
		}
		String[] n = arrPoint[index].split(",");
		double timehalf_ratio = (Double.valueOf(n[2]) - Double.valueOf(s[2])) /
(Double.valueOf(e[2]) - Double.valueOf(s[2]));
```

```
		features[12] = timehalf_ratio;
		//14.最大加速度
		double maxA = 0;
		for(int i = 0;i < arrPoint.length-2;i++) {
				String[] s9 = arrPoint[i].split(",");
				String[] e9 = arrPoint[i + 1].split(",");
				String[] f9 = arrPoint[i + 2].split(",");
			    double le1 = Math.pow(Double.valueOf(e9[0]) - Double.valueOf(s9[0]),
2) +
					Math.pow(Double.valueOf(e9[1]) - Double.valueOf(s9[1]), 2);
			    double le2 = Math.pow(Double.valueOf(f9[0]) - Double.valueOf(e9[0]),
2) +
					Math.pow(Double.valueOf(f9[1]) - Double.valueOf(e9[1]), 2);
				double v1 = le1 / (Double.valueOf(e9[2]) - Double.valueOf(s9[2]));
				double v2 = le2 / (Double.valueOf(f9[2]) - Double.valueOf(e9[2]));
				double a1 = (v2-v1) / (Double.valueOf(f9[2]) - Double.valueOf(s9[2]));
				if(a1 > maxA)
				{
					maxA = a1;
				}
			}
		features[13] = maxA;
		//15.是否停顿
		double stop = 0;
		for(int i = 0;i < arrPoint.length-1;i++) {
			String[] s10 = arrPoint[i].split(",");
			String[] e10 = arrPoint[i + 1].split(",");
			if(Double.valueOf(e10[0])== Double.valueOf(s10[0]) &&
Double.valueOf(e10[1])== Double.valueOf(s10[1])) {
				stop = 1;
			}
		}
		features[14] = stop;
		//16.最大速度
		double max_speed = 0;
		for(int i=0;i<speed.length;i++)
		{
			if(speed[i] > max_speed) {
				max_speed = speed[i];
			}
		}
```

```
features[15] = max_speed;
//17.最大角度
double maxM = 0;
	for(int i = 0;i < arrPoint.length-2;i++) {
		String[] s11 = arrPoint[i].split(",");
		String[] e11 = arrPoint[i + 1].split(",");
		String[] f11 = arrPoint[i + 2].split(",");
		double PI_A1 = 3.1415926535897;
		double x5 = Double.valueOf(s11[0]) - Double.valueOf(e11[0]);
		double y5 = Double.valueOf(s11[1]) - Double.valueOf(e11[1]);
		double x6 = Double.valueOf(f11[0]) - Double.valueOf(e11[0]);
		double y6 = Double.valueOf(f11[0]) - Double.valueOf(e11[0]);
		double c2 = (x5 * x6) + (y5 * y6);
		double z5 = Math.sqrt(x5*x5+y5*y5);
		double z6 = Math.sqrt(x6*x6+y6*y6);
		double A2 = c2/(z5*z6);
		double offset1 = Math.acos(A2)*180/PI_A1;
		if(offset1 > maxM) {
			maxM = offset1;
		}
	}
	features[16] = maxM;
//18.相邻点构成的角度序列一阶差分后的均值
	double aveFirst = 0;
	double sunFirst = 0;
	for(int i = 0;i < arrPoint.length-3;i++) {
		String[] s12 = arrPoint[i].split(",");
		String[] e12 = arrPoint[i + 1].split(",");
		String[] f12 = arrPoint[i + 2].split(",");
		double PI_A2 = 3.1415926535897;
		double x7 = Double.valueOf(s12[0]) - Double.valueOf(e12[0]);
		double y7 = Double.valueOf(s12[1]) - Double.valueOf(e12[1]);
		double x8 = Double.valueOf(f12[0]) - Double.valueOf(e12[0]);
		double y8 = Double.valueOf(f12[0]) - Double.valueOf(e12[0]);
		double c3 = (x7 * x8) + (y7 * y8);
		double z7 = Math.sqrt(x7*x7+y7*y7);
		double z8 = Math.sqrt(x8*x8+y8*y8);
		double A3 = c3/(z7*z8);
		double offset2 = Math.acos(A3)*180/PI_A2;

		String[] s13 = arrPoint[i+1].split(",");
		String[] e13 = arrPoint[i + 2].split(",");
```

```
                String[] f13 = arrPoint[i + 3].split(",");
                double PI_A3 = 3.1415926535897;
                double x9 = Double.valueOf(s13[0]) - Double.valueOf(e13[0]);
                double y9 = Double.valueOf(s13[1]) - Double.valueOf(e13[1]);
                double x10 = Double.valueOf(f13[0]) - Double.valueOf(e13[0]);
                double y10 = Double.valueOf(f13[0]) - Double.valueOf(e13[0]);
                double c4 = (x9 * x10) + (y9 * y10);
                double z9 = Math.sqrt(x9*x9+y9*y9);
                double z10 = Math.sqrt(x10*x10+y10*y10);
                double A4 = c4/(z9*z10);
                double offset3 = Math.acos(A4)*180/PI_A3;

                sunFirst += (offset3 - offset2);
            }
            aveFirst = sunFirst/ arrPoint.length-3;
            features[17] = aveFirst;

        return new LabeledPoint(label,Vectors.dense(features));
    });
    return labeledPoint;

}
private JavaRDD<LabeledPoint> testingPre(JavaRDD<String> tnf) throws Exception{
    JavaRDD<LabeledPoint> labeledPoint2 = tnf.map(f -> {
    double[] features = new double[18];
    String[] arr = f.split(" ");
    int label = Integer.valueOf(arr[0].trim());
    double path = 0;
    double timeSum = 0;
    String[] arrPoint = arr[1].split(";");
    double speed[] = new double[arrPoint.length-1];
    for(int i = 0;i < arrPoint.length-1;i++)
    {
        String[] s = arrPoint[i].split(",");
        String[] e = arrPoint[i + 1].split(",");
        //整个路程的长度

        double le = Math.pow(Double.valueOf(e[0]) - Double.valueOf(s[0]), 2) +
                Math.pow(Double.valueOf(e[1]) - Double.valueOf(s[1]), 2);
        path += Math.sqrt(le);
        //每一段的速度
        speed[i] = le /    (Double.valueOf(e[2]) - Double.valueOf(s[2]));
        //差分之和
        double reduce = Double.valueOf(e[2]) - Double.valueOf(s[2]);
        timeSum += reduce;

    }

    //1.整个路程的平均速度
    String[] s = arrPoint[0].split(",");
    String[] e =arrPoint[arrPoint.length - 1].split(",");
    double v = path / (Double.valueOf(e[2]) - Double.valueOf(s[2]));
    features[0] = v;
    //2.速度的方差
    double sv=0;
    for(int i = 0;i < speed.length;i++) {
        double l =Math.pow(Double.valueOf(speed[i]) - v, 2);
        sv += l;
    }
    double variance = sv / speed.length;
    features[1] = variance;
    //3.轨迹x方向走一半花的时间比
    double pathX = Double.valueOf(e[0]) - Double.valueOf(s[0]);
    double middleX = pathX / 2;
    double closeX = Math.abs(middleX - Double.valueOf(s[0]));
    int index = 0;
    for(int i = 0;i < arrPoint.length-1;i++) {
        String[] p = arrPoint[i].split(",");
        double x = Math.abs(middleX - Double.valueOf(p[0]));
        if(x < closeX)
        {
            closeX = x;
            index = i;
        }
    }
    String[] m = arrPoint[index].split(",");
    double x_ratio = (Double.valueOf(m[2]) - Double.valueOf(s[2])) /
(Double.valueOf(e[2]) - Double.valueOf(s[2]));
    features[2] = x_ratio;
    //4.停顿时间占总时间
    double timeCount = 0;
    for(int i = 0;i < arrPoint.length-1;i++) {
        String[] s1 = arrPoint[i].split(",");
        String[] e1 = arrPoint[i + 1].split(",");
```

```
                if(Double.valueOf(e1[0])== Double.valueOf(s1[0]) &&
Double.valueOf(e1[1])== Double.valueOf(s1[1])) {
                    timeCount += Double.valueOf(e1[2]) - Double.valueOf(s1[2]);
                }
            }
            features[3] = timeCount/(Double.valueOf(e[2]) - Double.valueOf(s[2]));
            //5.x坐标回退次数占总次数
            double back = 0;
            for(int i = 0;i < arrPoint.length-1;i++) {
                String[] s2 = arrPoint[i].split(",");
                String[] e2 = arrPoint[i + 1].split(",");
                if(Double.valueOf(e2[0]) < Double.valueOf(s2[0])) {
                    back++;
                }
            }
            features[4] = back/(arrPoint.length-1);
            //6.时间轴t差分后的均值
            double ave = timeSum/(arrPoint.length-2);
            features[5] = ave;
            //7.t进行一阶差分后的标准差
            double sumRe = 0;
            for(int i = 0;i < arrPoint.length-2;i++) {
                String[] s3 = arrPoint[i].split(",");
                String[] e3 = arrPoint[i + 1].split(",");
                double reduce = Double.valueOf(e3[2]) - Double.valueOf(s3[2]);
                double reducePow = Math.pow(reduce - ave, 2);
                sumRe += reducePow;
            }
            double stan = Math.sqrt(sumRe / (arrPoint.length-2));
            features[6] = stan;
            //8.平均加速度
            double sumA = 0;
            for(int i = 0;i < arrPoint.length-2;i++) {
                String[] s4 = arrPoint[i].split(",");
                String[] e4 = arrPoint[i + 1].split(",");
                String[] f4 = arrPoint[i + 2].split(",");
                double le1 = Math.pow(Double.valueOf(e4[0]) - Double.valueOf(s4[0]), 2) +
                        Math.pow(Double.valueOf(e4[1]) - Double.valueOf(s4[1]), 2);
                double le2 = Math.pow(Double.valueOf(f4[0]) - Double.valueOf(e4[0]), 2) +
                        Math.pow(Double.valueOf(f4[1]) - Double.valueOf(e4[1]), 2);
                double v1 = le1 / (Double.valueOf(e4[2]) - Double.valueOf(s4[2]));
                double v2 = le2 / (Double.valueOf(f4[2]) - Double.valueOf(e4[2]));
```

```
        sumA += (v2-v1) / (Double.valueOf(f4[2]) - Double.valueOf(s4[2]));
    }
    double avA = sumA / (arrPoint.length-2);
    features[7] = avA;
    //9.平均角度偏移和
    double offsets = 0;
    for(int i = 0;i < arrPoint.length-2;i++) {
        String[] s5 = arrPoint[i].split(",");
        String[] e5 = arrPoint[i + 1].split(",");
        String[] f5 = arrPoint[i + 2].split(",");
        double PI = 3.1415926535897;
        double x1 = Double.valueOf(s5[0]) - Double.valueOf(e5[0]);
        double y1 = Double.valueOf(s5[1]) - Double.valueOf(e5[1]);
        double x2 = Double.valueOf(f5[0]) - Double.valueOf(e5[0]);
        double y2 = Double.valueOf(f5[0]) - Double.valueOf(e5[0]);
        double c = (x1 * x2) + (y1 * y2);
        double z1 = Math.sqrt(x1*x1+y1*y1);
        double z2 = Math.sqrt(x2*x2+y2*y2);
        double A = c/(z1*z2);
        double offset = Math.acos(A)*180/PI;
        offsets += offset;
    }
    double aveOffset = offsets / (arrPoint.length-2);
    features[8] = aveOffset;
    //10.平均角度偏移标准差
    double sumOff = 0;
    for(int i = 0;i < arrPoint.length-2;i++) {
        String[] s6 = arrPoint[i].split(",");
        String[] e6 = arrPoint[i + 1].split(",");
        String[] f6 = arrPoint[i + 2].split(",");
        double PI_A = 3.1415926535897;
        double x3 = Double.valueOf(s6[0]) - Double.valueOf(e6[0]);
        double y3 = Double.valueOf(s6[1]) - Double.valueOf(e6[1]);
        double x4 = Double.valueOf(f6[0]) - Double.valueOf(e6[0]);
        double y4 = Double.valueOf(f6[0]) - Double.valueOf(e6[0]);
        double c1 = (x3 * x4) + (y3 * y4);
        double z3 = Math.sqrt(x3*x3+y3*y3);
        double z4 = Math.sqrt(x4*x4+y4*y4);
        double A1 = c1/(z3*z4);
        double offset = Math.acos(A1)*180/PI_A;
        double subPow = Math.pow(offset - aveOffset, 2);
        sumOff += subPow;
```

```
		}
		double offStand = Math.sqrt(sumOff / (arrPoint.length-2));
		features[9] = offStand;
		//11.速度的一阶差分平均值
		double sumSub = 0;
		for(int i = 0;i < speed.length-1;i++)
		{
			double sub = speed[i+1] - speed[i];
			sumSub += sub;
		}
		double aveSub = sumSub / (speed.length-1);
		features[10] = aveSub;
		//12.同一时间坐标是否变化
		double isStop = 0;
		for(int i = 0;i < arrPoint.length-1;i++) {
			String[] s7 = arrPoint[i].split(",");
			String[] e7 = arrPoint[i + 1].split(",");

	if((Double.valueOf(s7[0])!=Double.valueOf(e7[0])||Double.valueOf(s7[1])!=Double.valu
eOf(e7[1]))
					&&Double.valueOf(s7[2])==Double.valueOf(e7[2])) {
				isStop = 1;
			}
		}
		features[11] = isStop;
		//13.走一半路花的时间占总时间比
		double half_path = path/2;
		double path2 = 0;
		double min_diff_path = 100000;
		int index2 = 0;
		for(int i=0;i< arrPoint.length-1;i++)
		{
			String[] s8 = arrPoint[i].split(",");
			String[] e8 = arrPoint[i + 1].split(",");


			double l = Math.pow(Double.valueOf(e8[0]) - Double.valueOf(s8[0]), 2) +
					Math.pow(Double.valueOf(e8[1]) - Double.valueOf(s8[1]), 2);
			path2 += Math.sqrt(l);
			if(Math.abs(half_path - path2) < min_diff_path) {
				min_diff_path = Math.abs(half_path-path2);
				index = i;
```

```
            }

        }
        String[] n = arrPoint[index].split(",");
        double timehalf_ratio = (Double.valueOf(n[2]) - Double.valueOf(s[2])) /
(Double.valueOf(e[2]) - Double.valueOf(s[2]));

        features[12] = timehalf_ratio;
        //14.最大加速度
        double maxA = 0;
        for(int i = 0;i < arrPoint.length-2;i++) {
            String[] s9 = arrPoint[i].split(",");
            String[] e9 = arrPoint[i + 1].split(",");
            String[] f9 = arrPoint[i + 2].split(",");
            double le1 = Math.pow(Double.valueOf(e9[0]) - Double.valueOf(s9[0]), 2) +
                    Math.pow(Double.valueOf(e9[1]) - Double.valueOf(s9[1]), 2);
            double le2 = Math.pow(Double.valueOf(f9[0]) - Double.valueOf(e9[0]), 2) +
                    Math.pow(Double.valueOf(f9[1]) - Double.valueOf(e9[1]), 2);
            double v1 = le1 / (Double.valueOf(e9[2]) - Double.valueOf(s9[2]));
            double v2 = le2 / (Double.valueOf(f9[2]) - Double.valueOf(e9[2]));
            double a1 = (v2-v1) / (Double.valueOf(f9[2]) - Double.valueOf(s9[2]));
            if(a1 > maxA)
            {
                maxA = a1;
            }
        }
        features[13] = maxA;
        //15.是否停顿
        double stop = 0;
        for(int i = 0;i < arrPoint.length-1;i++) {
            String[] s10 = arrPoint[i].split(",");
            String[] e10 = arrPoint[i + 1].split(",");
            if(Double.valueOf(e10[0])== Double.valueOf(s10[0]) &&
Double.valueOf(e10[1])== Double.valueOf(s10[1])) {
                stop = 1;
            }
        }
        features[14] = stop;
        //16.最大速度
        double max_speed = 0;
        for(int i=0;i<speed.length;i++)
        {
```

```
        if(speed[i] > max_speed) {
            max_speed = speed[i];
        }
    }
    features[15] = max_speed;
    //17.最大角度
    double maxM = 0;
    for(int i = 0;i < arrPoint.length-2;i++) {
        String[] s11 = arrPoint[i].split(",");
        String[] e11 = arrPoint[i + 1].split(",");
        String[] f11 = arrPoint[i + 2].split(",");
        double PI_A1 = 3.1415926535897;
        double x5 = Double.valueOf(s11[0]) - Double.valueOf(e11[0]);
        double y5 = Double.valueOf(s11[1]) - Double.valueOf(e11[1]);
        double x6 = Double.valueOf(f11[0]) - Double.valueOf(e11[0]);
        double y6 = Double.valueOf(f11[0]) - Double.valueOf(e11[0]);
        double c2 = (x5 * x6) + (y5 * y6);
        double z5 = Math.sqrt(x5*x5+y5*y5);
        double z6 = Math.sqrt(x6*x6+y6*y6);
        double A2 = c2/(z5*z6);
        double offset1 = Math.acos(A2)*180/PI_A1;
        if(offset1 > maxM) {
            maxM = offset1;
        }
    }
    features[16] = maxM;
   //18.相邻点构成的角度序列一阶差分后的均值
    double aveFirst = 0;
    double sunFirst = 0;
    for(int i = 0;i < arrPoint.length-3;i++) {
        String[] s12 = arrPoint[i].split(",");
        String[] e12 = arrPoint[i + 1].split(",");
        String[] f12 = arrPoint[i + 2].split(",");
        double PI_A2 = 3.1415926535897;
        double x7 = Double.valueOf(s12[0]) - Double.valueOf(e12[0]);
        double y7 = Double.valueOf(s12[1]) - Double.valueOf(e12[1]);
        double x8 = Double.valueOf(f12[0]) - Double.valueOf(e12[0]);
        double y8 = Double.valueOf(f12[0]) - Double.valueOf(e12[0]);
        double c3 = (x7 * x8) + (y7 * y8);
        double z7 = Math.sqrt(x7*x7+y7*y7);
        double z8 = Math.sqrt(x8*x8+y8*y8);
        double A3 = c3/(z7*z8);
```

```
            double offset2 = Math.acos(A3)*180/PI_A2;

            String[] s13 = arrPoint[i+1].split(",");
            String[] e13 = arrPoint[i + 2].split(",");
            String[] f13 = arrPoint[i + 3].split(",");
            double PI_A3 = 3.1415926535897;
            double x9 = Double.valueOf(s13[0]) - Double.valueOf(e13[0]);
            double y9 = Double.valueOf(s13[1]) - Double.valueOf(e13[1]);
            double x10 = Double.valueOf(f13[0]) - Double.valueOf(e13[0]);
            double y10 = Double.valueOf(f13[0]) - Double.valueOf(e13[0]);
            double c4 = (x9 * x10) + (y9 * y10);
            double z9 = Math.sqrt(x9*x9+y9*y9);
            double z10 = Math.sqrt(x10*x10+y10*y10);
            double A4 = c4/(z9*z10);
            double offset3 = Math.acos(A4)*180/PI_A3;

            sunFirst += (offset3 - offset2);
        }
        aveFirst = sunFirst/ arrPoint.length-3;
        features[17] = aveFirst;
        return new LabeledPoint(label,Vectors.dense(features));
        });
        return labeledPoint2;
    }

}
```