|  |
| --- |
|  |
| Data Cleaning |

Revision Control

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version No | Date | Change / Activity | By | Approver | Circulation |
| PA1 | 31-Dec-18 | Initial Draft | Saurabh | Rajeev | Internal |
| A |  |  |  |  |  |

Reference Documents

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Document Name | Version / Date | Owner |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Copyright: SENSORISE 2018

SENSORISE Confidential and Proprietary Information

All Rights Reserved

The contents of this document are restricted for circulation; please use pursuant to company instructions and the terms of the Non-Disclosure Agreement.

Table of Contents

[1 Introduction 3](#_Toc1391324)

[2](#_Toc1391325) [[Mark point 6](#_Toc1391325)](#_Toc1391331)

[[High Level Description 6](#_Toc1391325)](#_Toc1391332)

[[Input 6](#_Toc1391325)](#_Toc1391333)

[[Output 7](#_Toc1391325)](#_Toc1391334)

[[Algorithm 7](#_Toc1391325)](#_Toc1391335)

[3 Algorithm used:](#_Toc1391325)

3.1 Entity Segregation  
[High Level Description 3](#_Toc1391326)

[Input 4](#_Toc1391327)

[Output 4](#_Toc1391328)

[Algorithm Description 5](#_Toc1391329)

[Algorithm 5](#_Toc1391330)

3.1 Path Marker  
[High Level Description 6](#_Toc1391343)

[Input 6](#_Toc1391344)

[Output 7](#_Toc1391345)

[Algorithm Description 7](#_Toc1391346)

[Algorithm 7](#_Toc1391347)

[4 Enhanced RDP Algorithm 7](#_Toc1391336)

[High Level Description 7](#_Toc1391337)

[Input 7](#_Toc1391338)

[Output 8](#_Toc1391339)

[Algorithm Description 8](#_Toc1391340)

[Algorithm 8](#_Toc1391341)

[5 FindSegmentPoint 9](#_Toc1391342)

[High Level Description 9](#_Toc1391343)

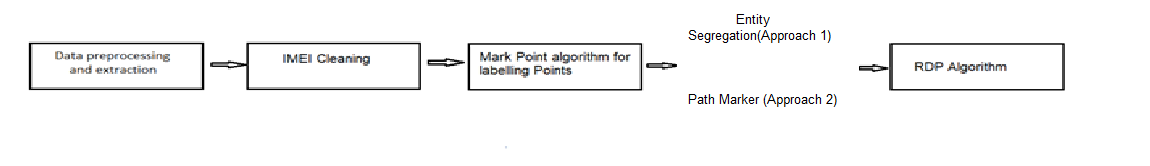
[Input 9](#_Toc1391344)

[Output 9](#_Toc1391345)

[Algorithm Description 9](#_Toc1391346)

[Algorithm 9](#_Toc1391347)

# Introduction

This document explains the suites of algorithms used to clean the Lat long data. It has been noticed that several times the lat long data received from the device contains

1. Invalid Lat Long, for example out of bound Lat or Long
2. Spurious data, for example a vehicle can not move 100 KM in less than a minute and come back to original track.
3. Duplicate Lat Long information when vehicle is in stationary state

Algorithms explained in the documents are

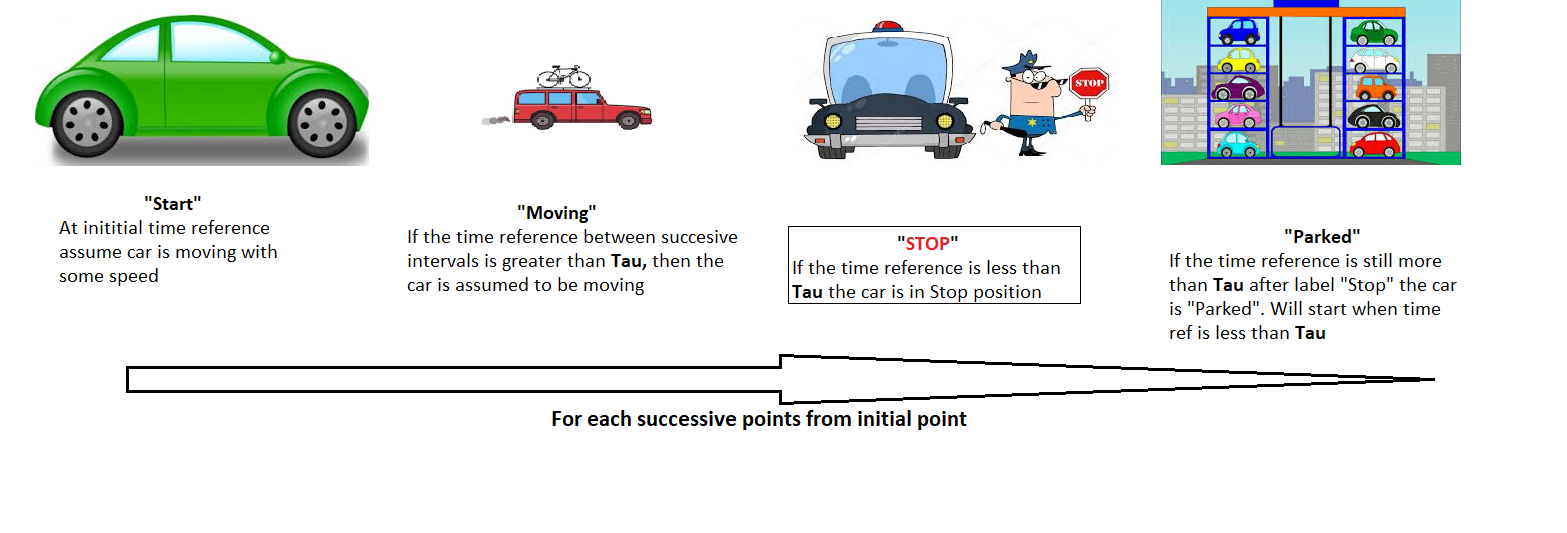
1. Mark Point algorithm which labels the data points movement with respect to vehicle
2. Entity Segregation algorithm which clusters the data points into two different points
3. Enhanced RDP Algorithm which removes redundant and duplicate data points

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Input | Output | Description |
| IMEI Cleaning | Data after IMEI Extraction | Cleaned data removing all duplicate points and stationary points | All Duplicate points and stationary points are removed from the IMEI extracted file |
| Mark Point | Data after IMEI Cleaning | New Columns:   * **TripID**: Columns with integer value * **Remove**: Column with either “0”, “Static Point” and “Duplicate Points” | This algorithm marks trips for different data points with input parameters so that we could get identify in how many trips has the vehicle moved |
| Separate Segments | Data After mark point | New Columns:  **SegmentID**: Columns with integer value | The algorithm helps us to identify spurious points in data and accordingly find out how many different paths were there in different trips of data |
| Modified RDP | Data after Segment Points | New Columns:  **RDP**: Columns with integer value “0”, “1”and “-1” | This algorithm removes point which are less deviated from the initial path based on speed an distance  RDP Value :  0 : It means the point has less deviated from the path  1: It means the point is valid point for analysis  -1 : It means the point is stationary. |

# Mark point

## 1.High Level Description

1. After sorting the data points after clustering we get data points showing vehicle path and route for each cluster. Of all these points we need to point out points where the vehicle has started moving and stopped for analysis of driver behavior to be done correctly.
2. This algorithm does the same by using time reference we would assume if the vehicle has stopped moving the time difference between new and previous point would be more than our input variable Tau(just say 30 minutes as reference)
3. In this way we can label all the points in the data with three labels ”Start”, “Moving”, ”Parked” and “Stop” or Trip ID for path marking



## 2.Input

This requires three inputs

* Table with two columns Lat and Long coordinates for clustering
* Date time for time calculation
* Static Dist for static point removal
* Tau for time reference
* Big Tau for time reference
* Speed Trip for speed reference

## 3.Output

Parameters which are returned

* Column value Trip ID with different integer value for each new trip of data.
* Column value Remove with values “0”,”Static Points”, “Duplicate Points” for points marking

## 4..Algorithm

1. For each consecutive data point calculate the distance with respect to previous data index, we require to check where the vehicle has been stationary or not
2. If the static distance is zero(say) , we would mark the previous point as Duplicate and remove the point from our data
3. Arrange and sort the value with date time for the whole data. Points which are having initial date time will be in initial position followed by later points
4. Now calculate the distance again with respect to previous data point and calculate the static dist
5. If the static distance is less than Static Dist ~ 35 meters(say) , we would mark the previous point as Stationary and remove the point from our data
6. For each consecutive data point calculate the time with respect to previous data point’s time index, we require to check where the vehicle has stopped moving
7. If the respective time is more than particular time(bigTau), we will assume the vehicle has stopped moving and mark the point with different TripID and continue with remaining points
8. If the reference time is more than (Tau) then check for speed w.r.t to previous points. If the speed is less than reference speed(speed\_trip) we would assume that point to be of different trip
9. Repeat all the steps until all data points in the column is been finished

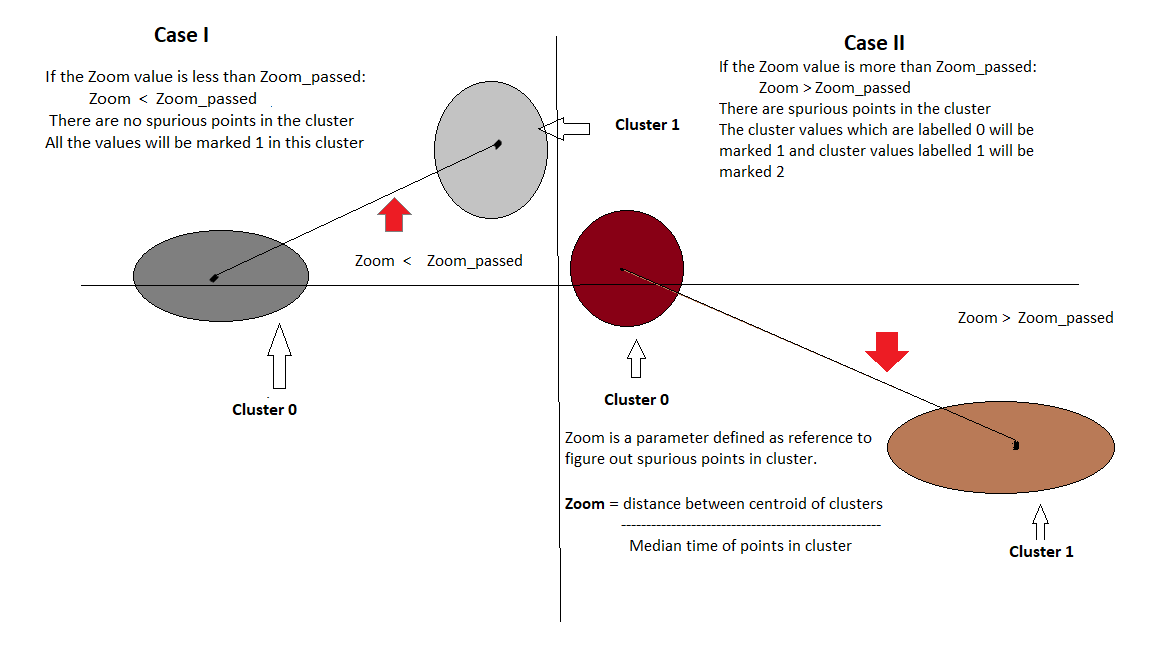
# Algorithm for Data Cleaning

## Introduction:

* The reason behind deciding algorithm approach is to separate out data points which are spurious from the set of points in lat long data
* The algorithm had to separate out points and hence get the path which the vehicle travelled during its commutation
* Two different approaches are established for cleaning the data and getting the points which are discussed in detail

# Entity Segregation(Approach 1- Not Used)

## 1.High Level Description

* The reason behind using this algorithm is spurious and inconsistent data points i.e in data points which are geographically located very far away inconsistently, in respective same time interval can be termed as spurious points as a vehicle with constant speed and in limited amount of time can travel distances which should be in particular range.
* Data is sent by both server as well as the vehicle, so we needed a particular way to separate out these set of data for our analysis of driver behavior to be done correctly
* For each hour of data, we need to calculate and cluster out the points so that a definite path exists for vehicle movement
* For this analysis we have used K-means clustering algorithm( Details of algorithm in algorithm description)
* After the application of this algorithm we get datapoints which will be indexed as “1” or “0”
* All the cluster data points having values labelled as “1” and labelled “0” will have alternate different points which will be considered in a particular path. 
* Zoom value is taken here for reference to remove spurious data points in the cluster
* Zoom value is defined as distance between centroids of two clusters and median time difference between points of cluster

## 2.Input

This requires two inputs

* Table with two columns Lat and Long coordinates for clustering
* Date time for time calculation
* Zoom\_passed(Explained below)

## 3.Output

Parameters which are returned

* A new column “Cluster” with value 1 and 2 which will be marked as per cluster from a vehicle’s start point to stop point.

## 4.Algorithm Description

* The main algorithm used here is K-Means clustering algorithm:
* This algorithm is used to classify a given set of points into certain number of clusters(in this case we are using two cluster point which will be labelled as 0 and 1)

1. Initially randomly choose (K = 2)-Points at a point as initial centroids
2. Cluster Assignment: The data points, that are similar closet to the center i.e Euclidean distance is less is considered as one cluster
3. Move the cluster centroid value to new set of data points. The new centroid value will be mean of all data points in the particular cluster
4. Do until Euclidean distance of new center with the old cluster center is not zero

## 5.Algorithm

1. Sort the data into timely data points for calculating distances and applying K-Means algorithm
2. For each hour of data from starting time frame stamp to next hour of data:
   1. Create two clusters between the points of the data for the first hour
   2. Now calculate centroid distance between the clusters so that we can calculate and remove the spurious entities into new cluster
   3. Find the average time in each cluster entity so that we can check how vehicle has travelled in particular instant of time
      1. Calculate **Zoom** which is distance between two cluster centers divided by difference between mean time of two clusters

Zoom = distance between two cluster centers / median time difference between two clusters

* + 1. If the Zoom < Zoom\_passed(Passed value in function)
       1. Mark label all the points as 0
    2. Else
       1. The clustered points labeled by K-Means are labeled in that order only

1. Repeat it for the data points from the second hour of data until all data points are over with an exception of clustering i.e

**While clustering from second hour of data cluster the points clustered in the current hour of data starting with the last point of previous hour of data**

# Separate Segments(Path Marker - Approach 2)(Executed In This Currently)

## 1.High Level Description

* The initial hypothesis of having two clusters proved out to be incorrect while running clustering, so Path marker algorithm is used to segregate all the path within one segment (Start to stop within time frame)
* The initial hypothesis made here is within each segment there are ‘n’ number of paths present for a vehicle
* The paths can be separated by some reference distance and speed as a vehicle cannot be at two different location at same time stamp or could travel insignificant distances in small amount of time
* Therefore for each Segment there could be different paths for a vehicle which we need to segregate by algorithm for driver behaviour to be done correctly

## 2.Input

This requires two inputs

* Table with two columns Lat and Long coordinates for clustering
* Date time for time calculation
* Dist\_ref ,Speed\_ref1 and speed\_ref2

## 3.Output

Parameters which are returned

* A new column “SegmentID” with values which will be marked as per path from a vehicle’s one trip.

## 4.Algorithm

Take each segment from start to stop label i.e one trip ID marked after mark point algorithm

For each segment of data from starting time frame stamp to stop point of data:

* 1. Assign mark point as label “0” for first point of data
  2. Now calculate distance and speed for each point of data with respect to last data point of marked label iteratively:
* If the speed is more than reference value (ref\_speed1) update that point with the with different segmentID
* If the speed is more than reference value (ref\_speed2) and less than reference value (ref\_speed1)
  + Check the distance with respect to last point if less than reference distance(Dist\_ref):
    - update that point with the with different segmentID
* Else mark the point with same segment ID as the last point ID

# Enhanced RDP Algorithm

## High Level Description

This algorithm identifies redundant and duplicate data points by identifying those points which are not adding much value in the data. For example, if a vehicle is going in a straight line on a high way with minor deviations due to overtaking or driving skills or change of lane, every 10 Seconds data may not add value unless it has significantly different other attributes like Speed. So if a vehicle is driving at a constant speed (Standard Deviation is very small) of 50KM/Hr from at time 10:30 am to 11:30am, so the 360 points during this time can be reduced to 2 points one at 10:30 and other at 11:30.

## Input

This requires three inputs

1. Table with three columns Date/Time, Lat and Long
2. Max Deviation (Epsilon): Any deviation below this is treated as points in the straight line. Typical values are 2-5 meters, which could be the width of the highway
3. Speed Deviation (Zeta) : Standard deviation of the speed for the points to be removed should be below this

## Output

Array of integer indicating which points in the input table should be kept (1) or removed (0)

## Algorithm Description

Idea behind the algorithm is to find maximum deviation point (LatLongD) between two latlongs (Latlong1 and LatLong2). If the maximum deviation point is more than Epsilon or Variance in the speed (during the segment is more than Zeta), then, the data is split into two segments as follows

1. Segments1: LatLong1 to LatLongD
2. Segments2: LatLongD to latLong2

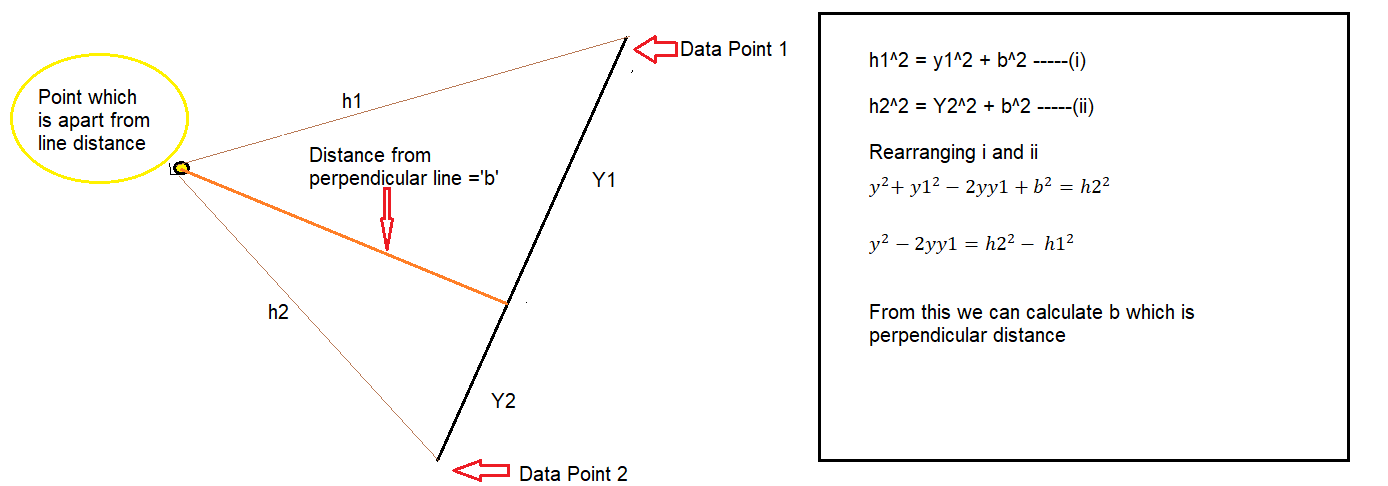
Repeat this for each sub-parts. If in a segment of data , maximum deviation is less than Epsilon then only 1st and last points in the segments are kept.

Normally the algorithm starts from 1st data point as LatLong1 and last data point as LatLong2.

## Algorithm

* Calculate distance of first point with respect to each other point, split the point values with respect to the maximum value and pass the values to the stack for application of RDP.
* Create a Stack of Segments (Index1 and Index2) and Push 0 as Index1 and Last index of the table as Index2.
* Initialize integer array RetainPoints with same dimension as the number of rows in the table and set all values to 0.
* Do Until there is a segment in the stack to be processed
  + Pop the Segment from the Stack
  + For each point between Index1 to Index2 find distance perpendicular to the line joining LatLong(index1) and LatLong(index2) (make a function FindSegmentPoint to do this)
    - This will return three parameters MaxIndex point, MaxDeviation and SD of Speed.
  + Check if SD(Speed) > Zeta OR MaxDeviation > Epsilon
    - Create two Segments and push them on the stack
      * Segment1 : Index1 to MaxIndex
      * Segment2: MaxIndex to Index2
  + ELSE
    - Mark RetainPoints(index1 + 1 to Index2-1) as 1

# FindSegmentPoint



## High Level Description

For each point between two indexes in the data, find distance perpendicular to the line joining LatLong(index1) and LatLong(index2). If there are 100 data points from Index1 to Index2, then there would be 98 distances. The perpendicular distance ‘b’ is the maximum deviation from line which we are calculating. If it is less than some tolerance level we will ignore else we will keep.

The maximum speed deviation is the speed of each successive data points with respect to last data point. If the deviation is more than the speed specified(Zeta) we need to keep those values in RDP as these high values cannot be ignored which is important parameter in predicting driver behaviour

## Input

This requires three inputs

1. Table with three columns Date/Time, Lat and Long
2. Index1, Index2

## Output

Three parameters are returned

1. Index of the maximum deviation point
2. Distance of the maximum deviation point
3. Standard Deviation of the speed

## Algorithm Description

As stated in the Section 3.1

## Algorithm

* B = 0 (Small number)
* MaxIndex = Index2
* For each index between index1 and index2
  + Using trigonometric formulation find the perpendicular distance of the point(@index) from the line joining Index1 and Index2
    - ---- 1
    - ----- 2
    - Where
      * h1 is the haversine distance of Point(@index) and Point(@Index1) and
      * h2 is the haversine distance of Point(@index) and Point(@Index2) and
      * y1 is the unknown point on the line between index1 and index2 where perpendicular falls. This is haversine distance from Point(@index1) to the perpendicular
      * y2 is the unknown point on the line between index1 and index2 where perpendicular falls. This is haversine distance from Point(@index2) to the perpendicular
      * y is the haversine distance between Point(@Index1) and Point(@Index2). This is also same as y1 + y2
      * b is the perpendicular distance which we need to find.
    - Rearranging equation 2 we get
      * --- 3
    - acting equation 3 and 1, we get
      * --- 4
    - Since in the equation 4, there is only one unknown y1, it can be easily found.
    - Knowing y1, we can find b using equation 1.
    - Check if it is greater than the previous maximum B if so then initialize B as b and MaxIndex as Index
  + Find Speed between Index and its previous Index
    - If Duration is 0 then assume speed to be 0
    - Speed = Distance between Index and Previous Index / Duration
* Find the Standard Deviation of all the speeds
* Return MaxIndex, B and StandardDeviation of Speed

# Procedure To Execute

* + - The IMEI Files whose data is to be processed are multi processed with four core processors with main file specifying function name
    - Files to be processed are kept in Data folder inside clean Clean Records sub folder
    - The directory has to be mentioned where files are to be stored in **directory** variable i.e in this case directory = "C:\\Users\\saura\\Documents\\Clean Records" in the main file of clean Data
    - Create a folder named IMEI\_Files where final output record in form of .csv file will be stored
    - Open the cmd command line where we have stored the python script of IMEI Extraction and type in cmd – python Main\_File\_IMEI\_Cleaning.py(name of file)

# Summary Files

1. **IMEI Heuristics**

* IMEI Heuristics gives summary of cleaned IMEI files
* IMEI Heuristics is generated with distance, speed and time heuristics as different files in specified directory mentioned by us in our specified directory,

dir = “C:\Users\saura\Documents\Clean Records\IMEI\_Heuristics”

1. **IMEI Description**

IMEI Description gives seven different files with different attributes which are following:

* Days\_Description : The number of days the vehicle has moved in different day of week
* Distance\_Description: Number of Kilometers each vehicle has travelled each day
* Hour\_Description: Time travelled in each hour by each vehicle during the day
* IMEI\_Description: General information about IMEI
* Segment\_Description: Total segments in total run of vehicles
* Time\_Description: Time travelled by each vehicle
* Trip\_Description : Total trips in total run of vehicles

1. **Summary Screen**

* Summary screen gives clean record summary of records in the files after IMEI cleaning for proper visualization
* IMEI Description is generated with seven different files in specified directory mentioned by us in our specified directory,

dir = “C:\Users\saura\Documents\Clean Records\IMEI\_Description”

