Distance_from_KML_map_data

1 Description

1.1 KML: Keyhole Markup Language

The Kml file used to display the geographical data in an earch bowser such as Google each. KML is an XML notation for expressing geographic annotation and visualization within 2D maps and 3D earth browsers.

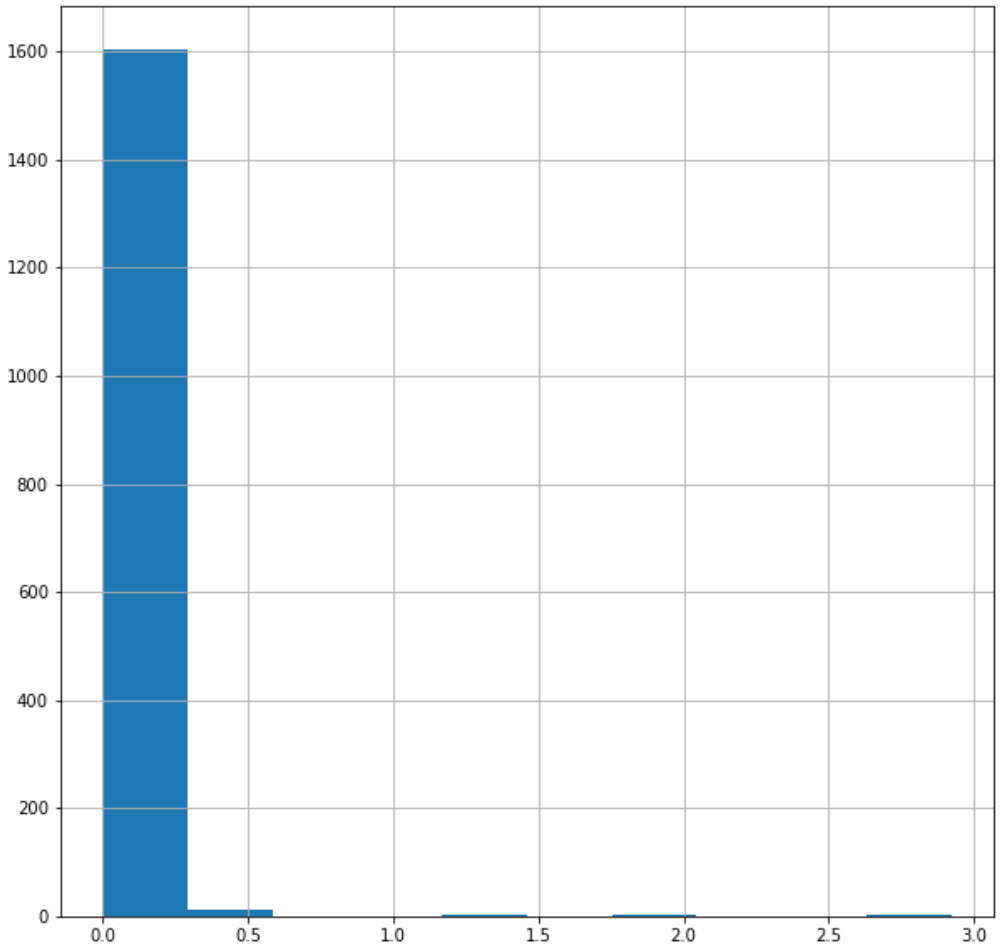
1.2 Problem statement

The provided KML map data describes the route of a vehicle. Since these are recorded using sensors, they might have incorrect measurements. The goal of this project is to read the given kml file, check out the coordinates, and with appropriate filtering logic remove the incorrect data (duplicates or outliers). And, finally calculate the distance between the coordinates and display the total distance travelled by hte vehicle in kilometers.

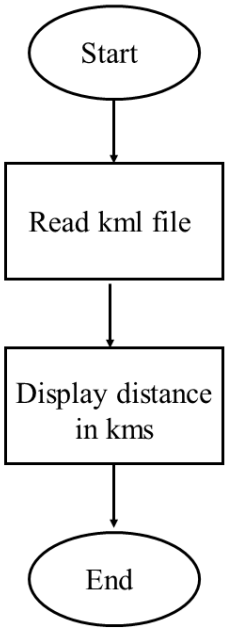
2 Filtering Logic

The sensors records the position of the vehicle continuously by storing coordinates at each position. It is observed that the distance between two cordinates approximately lies between 0-0.06km (statistical analysis: histogram). Any distance more than 50m considered as outliers or incorrect data points.

The file also containes a number of duplicate points which has no or little influence on the final distance measure and hence can be removed to optimise the calculation time.


1600
1400
1200
1000
800
600
400
200
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

3 Code



3.1 Requirements

- The requirements.txt file contains all the required dependencies for this project.
- python==3.70
- IDE used VS code(or your prefferred IDE)

3.2 Environment setup

- **Virtual environment creation**

  conda create -p distkml python==3.7 -y

- **Activate virtual environment**

  conda activate distkml

  OR

  conda activate distkml/

- **run requirements.txt for installing required libraries or packages**

  pip install -r requirements.txt

3.3 code strucure

- main.py : runs the whole projects by running it using run button or type "python main.py" on IDE termnal and click enter to run the project.

- dist_log.py: helps to create log files

- dist_exception.py: provides exception handling

- distance_kms.py: do all required tasks and return appropriate results.

- kml parsing.ipynb: jupyter notebook for analysis

- task_2_sensor: input kml file

3.4 git action

```git clone <git_repo_url>```

```git add .```

```git commit -m "message or comment"```

```git push origin main```