# Analysis and Inference Extraction of Car using Onboard Diagnostic(OBD) Data Report

## 1. Created By:

**Name:** Hrishikesh Shinde

## 2. Introduction

I this case study we have analyzed the sensor data of a car which have been logged using OBD and extracted the essential inferences. Here, the main inferences which have been extracted are:

a) High Fuel Consumption Points.
b) Pothole Detection.
c) Hard Breaking Detection.
d) Rash Driving Detection.
e) Engine Overheated Detection.

Further, we'll look one by one in detail.

## 3. Software Requirements

1. Python 3.x+.
2. Integrated Development Environment (IDE)
3. Data logged by OBD

Required Python Libraries:

1. **Folium**: To plot Points on Map.
   *Installation: pip install folium*

2. **Pandas**: To import dataset.
   *Installation: pip install pandas*

3. **Matplotlib**: To plot Graphs.
   *Installation: python -m pip install -U matplotlib*

4. **MPL_toolkits**: To plot 3 Dimensional Plot.
   *Installation: pip install --upgrade matplotlib*

5. **Sys:** Used for exit.
   *Installation: Pre-Installed*

## 4. Car's Sensor Data Logged Using OBD

The data is stored in excel spreadsheet which we'll be using to find inferences. The data which is logged look as below:

| 1 | GPS Time | Device Time | Longitude | Latitude | Acceleration Sensor(Total)(g) | Acceleration Sensor(X axis)(g) | Acceleration Sensor(Y axis)(g) | Acceleration Sensor(Z axis)(g) | Accelerator PedalPosition D(%) |
|---|---|---|---|---|---|---|---|---|---|
| 2 | Sat Nov 02 11:24:04 GMT+05:30 2019 | 24:04.6 | 78.53707726 | 17.34112692 | -0.02984698 | 0.11343722 | 0.53545415 | 0.70060915 | 14.90196133 |
| 3 | Sat Nov 02 11:24:05 GMT+05:30 2019 | 24:05.6 | 78.53708293 | 17.34116309 | 0.0031024 | 0.20928079 | 0.56927407 | 0.69853354 | 14.90196133 |
| 4 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:06.6 | 78.53708861 | 17.34119926 | 0.0750325 | 0.13272803 | 0.61457086 | 0.77154553 | 14.90196133 |
| 5 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:07.6 | 78.53709428 | 17.34123543 | 0.0597567 | -0.13538975 | 0.57916367 | 0.7766735 | 14.90196133 |
| 6 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:08.6 | 78.53709995 | 17.3412716 | 0.71212858 | -0.83767271 | -0.28330621 | 1.36540926 | 14.90196133 |
| 7 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:09.6 | 78.53710562 | 17.34130777 | 0.02663771 | -0.06872657 | -0.01152566 | 0.92391849 | 14.90196133 |
| 8 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:10.6 | 78.5371113 | 17.34134394 | 0.08770105 | 0.13297223 | -0.06182826 | 0.97739559 | 14.90196133 |
| 9 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:11.6 | 78.53711697 | 17.34138011 | 0.02658044 | -0.07898244 | -0.00004885 | 0.92318594 | 14.50980377 |
| 10 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:12.6 | 78.53712264 | 17.34141628 | 0.02219938 | -0.02794727 | -0.03130484 | 0.92098826 | 14.50980377 |
| 11 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:13.6 | 78.53712832 | 17.34145245 | 0.02117257 | -0.08728481 | 0.0028814 | 0.9170813 | 14.50980377 |
| 12 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:14.6 | 78.53713399 | 17.34148862 | 0.0666879 | -0.07385451 | -0.04058396 | 0.9629885 | 30.58823586 |
| 13 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:15.6 | 78.53713966 | 17.3415248 | 0.01193536 | -0.0274589 | -0.03130484 | 0.91073239 | 30.58823586 |
| 14 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:16.6 | 78.53714534 | 17.34156097 | -0.09632336 | 0.03871588 | -0.04156071 | 0.80158061 | 30.58823586 |
| 15 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:17.6 | 78.53715101 | 17.34159714 | 0.02382111 | 0.04506476 | -0.03936302 | 0.92172086 | 25.49019623 |
| 16 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:18.6 | 78.53715668 | 17.34163331 | 0.07383185 | 0.03896007 | -0.00200235 | 0.97275603 | 25.49019623 |
| 17 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:19.6 | 78.53716235 | 17.34166948 | -0.08473359 | 0.00501802 | -0.13117748 | 0.80548757 | 25.49019623 |
| 18 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:20.6 | 78.53716803 | 17.34170565 | -0.00467156 | 0.0443322 | -0.12140999 | 0.88655788 | 29.41176605 |
| 19 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:21.6 | 78.5371737 | 17.34174182 | -0.01331575 | -0.06384283 | -0.06451432 | 0.88216239 | 29.41176605 |
| 20 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:22.6 | 78.53717937 | 17.34177799 | 0.10347728 | -0.09607557 | -0.05230495 | 0.99766308 | 29.41176605 |
| 21 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:23.6 | 78.53718505 | 17.34181416 | 0.01278169 | 0.04066939 | -0.01763034 | 0.91146499 | 36.47058868 |
| 22 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:24.6 | 78.53719072 | 17.34185033 | 0.02353568 | -0.04894739 | 0.05562588 | 0.9204998 | 36.47058868 |
| 23 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:25.6 | 78.53719639 | 17.3418865 | 0.0284496 | -0.00719135 | -0.00273491 | 0.92806965 | 36.47058868 |
| 24 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:26.6 | 78.53720206 | 17.34192267 | 0.05911226 | -0.03161009 | -0.10334012 | 0.95322096 | 30.19607925 |
| 25 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:27.6 | 78.53720774 | 17.34195884 | -0.0354052 | -0.12098268 | 0.01533496 | 0.85652274 | 30.19607925 |
| 26 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:28.6 | 78.53721341 | 17.34199501 | 0.0294173 | -0.04552877 | -0.03130484 | 0.92758131 | 30.19607925 |
| 27 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:29.6 | 78.53721908 | 17.34203118 | -0.01369674 | -0.06604051 | -0.03472346 | 0.88313925 | 30.98039246 |
| 28 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:30.6 | 78.53722476 | 17.34206735 | -0.03437518 | -0.05407533 | -0.0183629 | 0.86360425 | 30.98039246 |
| 29 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:31.6 | 78.53723043 | 17.34210352 | 0.01725314 | -0.04699389 | -0.0203164 | 0.91561615 | 26.27450943 |
| 30 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:32.6 | 78.5372361 | 17.34213969 | -0.06910246 | -0.0972965 | -0.08722375 | 0.82135981 | 26.27450943 |
| 31 | Sat Nov 02 11:24:06 GMT+05:30 2019 | 24:33.6 | 78.53724177 | 17.34217586 | 0.06049369 | -0.07068007 | -0.02910715 | 0.95737219 | 26.27450943 |

This dataset have 20 columns and 2446 rows.

The columns which we've considered to find inferences are:

1. GPS Time.
2. Device Time
3. Longitude.
4. Latitude.
5. Acceleration Sensor(Total)(g).
6. Acceleration Sensor(X axis)(g)
7. Accelerator PedalPosition D(%)
8. Engine Coolant Temperature(Â°C)
9. Engine Load(%)
10. Engine RPM(rpm)
11. Fuel flow rate/minute(gal/min)
12. Throttle Position(Manifold)(%)
13. Trip Time(Since journey start)(s)

The analysis of the above columns lead to create 5 inferences.

**Changes in dataset:**

As per the dataset provided to us. The GPS was turned off for a long time the logged data contained repeated values of GPS. We only got 4 points. I have found the in between values of the points and connected the points. I have found all the latitude and longitude of the route from source to destination. Also, I have took permission from trainer.

The changed latitude and longitude dataset is provided in the code folder.

# 5. Inference Description.

1. **High Fuel Consumption Points Detection:**

   When driver is on max speed of the particular gear and not switching to next gear. He will be wasting more amount fuel. Pedal will be pressed more and fuel consumption at this condition will be high. So, here I have detected the point which are consuming more fuel under the above condition.
   **Thresholds:**
   Acceleration pedal position > 55,
   Fuel Consumption Rate > 0.029,
   Throttle Position > 80

2. **Pothole Detection**

   When driver drives through the pothole the accelerometer value changes rapidly. So I have detected the sudden change in the accelerometer value. If the value of accelerometer goes above the threshold value then the pothole is detected.
   **Threshold:** Acceleration Total (g) <-0.19

3. **Hard Breaking Detection:**

   This inference is detected when there is a sudden drop in Engine RPM and Engine Load. So by checking the consecutive values of the RPM and Load this inference is detected.
   **Threshold:**
   1) Difference between two consecutive Engine RPM should be more than -950
   2) Difference between two consecutive Engine Load should be more than -70

4. **Rash Driving Detection:**

   In this inference we'll check if the car is in motion or not using accelerometer X-axis value. The we'll check the Pedal Position and Engine RPM. If its beyond the threshold value then rash driving is detected.
   **Threshold:**
   Accelerator PedalPosition D(%)>58
   Engine RPM > 800
   Accelerometer X axis>=0

5. **Engine Overheat Detection:**

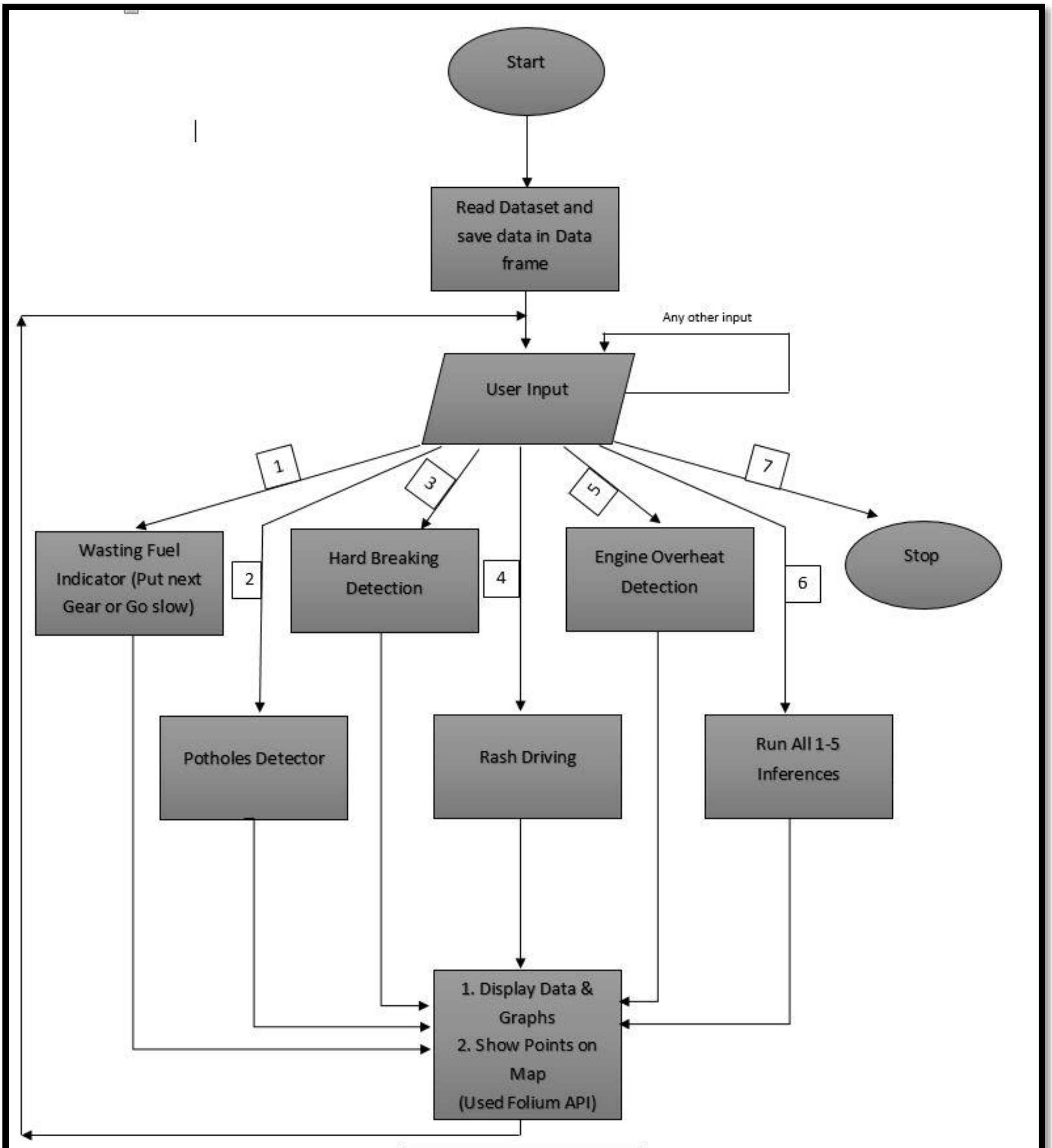
As the engine temperature increase the coolant temperature increase. We'll set a threshold value for engine coolant temperature. If the coolant temperature exceeds that then we'll detect the engine overheated point.

**Threshold:**

Coolant Temperature > 90°C

# 6. Flow of the Program.

The flow of the program is as show in the below activity diagram.




Activity Diagram (214966)

Exception handling of the user input is done in the program. Also, the program will stop only when user enter 7 as the input.

# 7. Algorithm

1. Read the logged data from excel file.
2. Show 5 inferences to user.

3. Ask for the user input.

   A. If input is equal to 1.
      Then, wasting fuel algorithm.
      Plot 3-D Graph of Trip Time(s), PedalPosition D(%), Fuel flow (gal/min)
      a. Check each entry if it's,
         Acceleration pedal position > 55
         Fuel Consumption Rate > 0.029
         Throttle Position > 80
         1. If all conditions satisfied,
            Plot the points on the 3-D graph.

      Using Latitude and Longitude of detected points, show Points on map.

   B. If input is equal to 2.
      Then, wasting fuel detection algorithm.
      Plot 2-D Graph of Trip Time(s), Acceleration Sensor(Total)(g).
      a. Check each entry if it's,
         Acceleration Total (g) <-0.19
         If above conditions is satisfied,
         1. Plot the points on the 2-D graph.

      Using Latitude and Longitude of detected points, show Points on map.

   C. If input is equal to 3.
      Then, Hard Braking Detection algorithm
      Plot 3-D Graph of Trip Time(s), Engine Load(%), Engine RPM(rpm).
      a. Check each entry if the,
         Difference between two consecutive RPM should be more than -950
         Difference between two consecutive Engine Load should be more than -70
         If above conditions are satisfied,
         1. Plot the points on the 3-D graph.

      Using Latitude and Longitude of detected points, show Points on map.

   D. If input is equal to 4.
      Then, Rash Driving Detection algorithm

Plot 2-D Graph of Trip Time(s), Accelerator PedalPosition D(%).

b. Check each entry if the,
   When Car is moving accelerometer's
   1) Positive values indicate an increase in velocity.
   2) Negative values indicate an decrease in velocity.
   3) Zero values indicate constant velocity

   Here we considered the positive value and zero value to confirm the vehicle is motion or not. Then we detected the value if pedal position. If both condition match rash driving is detected
   Engine RPM > 800
   If above conditions are satisfied,
   2. Plot the points on the 2-D graph.

Using Latitude and Longitude of detected points, show Points on map.

E. If input is equal to 5.
   Then, Engine Over Heat Detection algorithm
   Plot 2-D Graph of Trip Time(s), Engine Coolant Temperature(Â°C).
   c. Check each entry if the,
      Coolant Temperature > above 90°C
      If above conditions are satisfied,
      3. Plot the points on the 2-D graph.

   Using Latitude and Longitude of detected points, show Points on map.

F. If input is equal to 6
   Execute all above algorithms.
   Show map.

G. If input is equal to 7
   Stop Execution

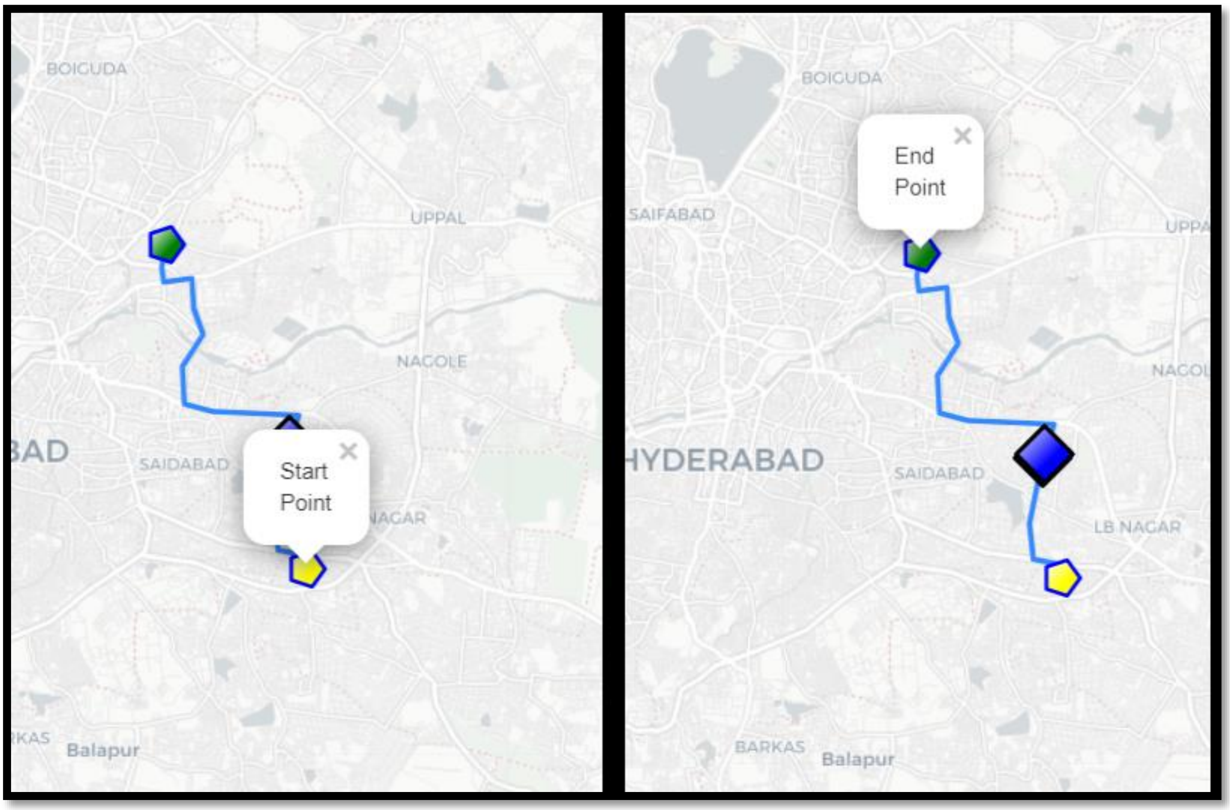
H. If any other Input:
   Repeat from step 3

--------------------------------------- End of the Algorithm ---------------------------------------

# 8. UI Interface

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
```

**Maps:** Click on the marker to view the information of point.



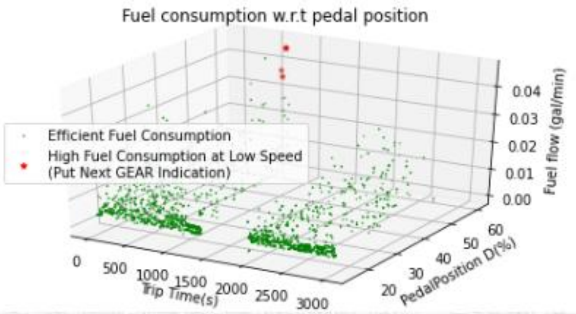
# 9. UI Artifacts

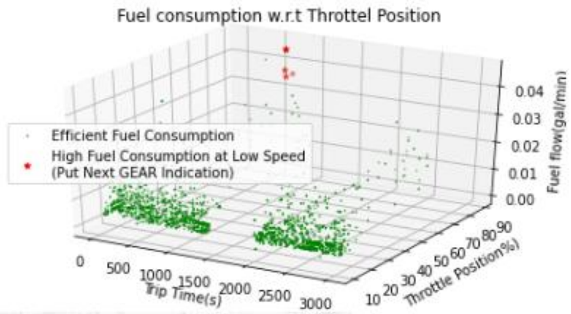
**User Input ‘1’:**

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
1
```

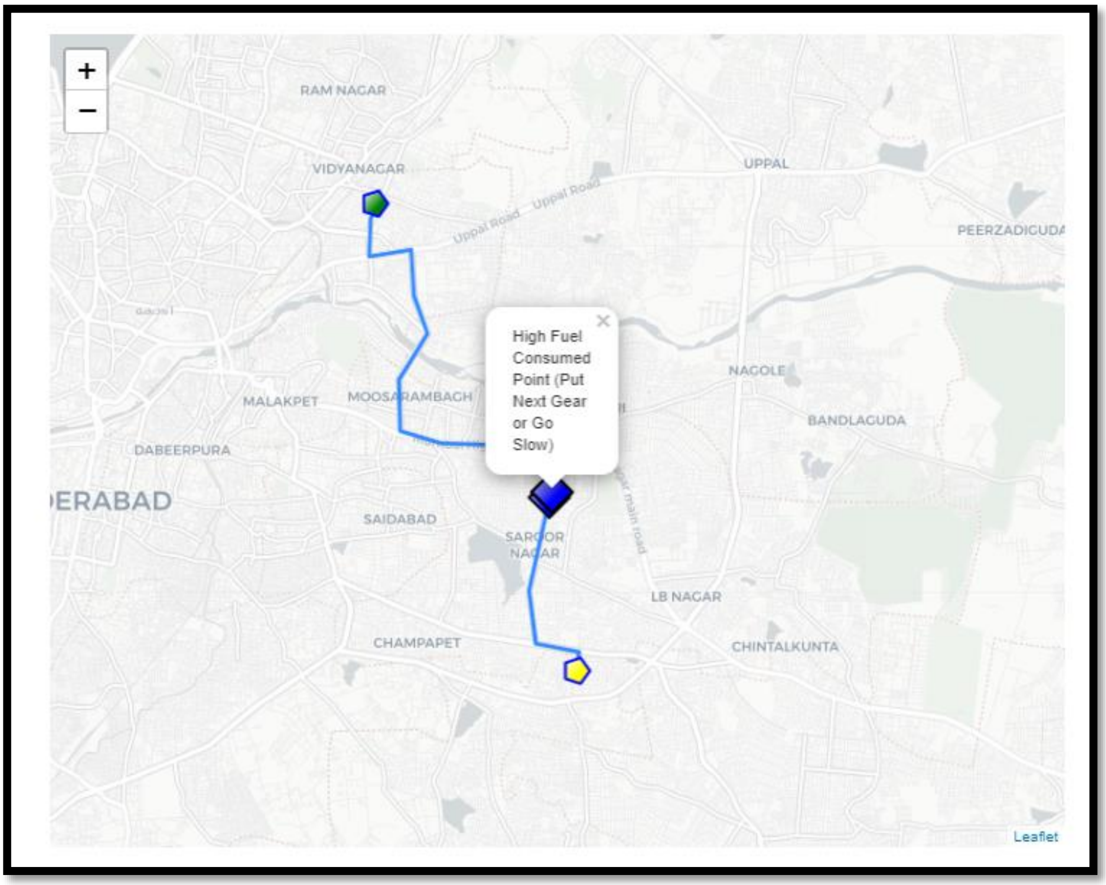
Output (Part 1):

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
1
----------------------- INDICATION TO PUT NEXT GEAR or GO SLOW ---------------------

------------------------- High Fuel Consuption Detected at -------------------------

 DEVICE TIME              LATITUDE                LONGITUDE               HIGH FUEL CONSUPTION

00:35:47.600000          17.36208677777778       78.53375592592592       0.03804309
00:36:17.600000          17.362721444444446      78.53398948148148       0.03603455
00:36:18.600000          17.3627426       78.53399726666666       0.03603455
00:36:19.600000          17.362763755555555      78.53400505185185       0.04577404

-------------------- High Fuel Consuption Detected 4 times ------------------------------
```



Output (Part 2):




Output (Part 3):

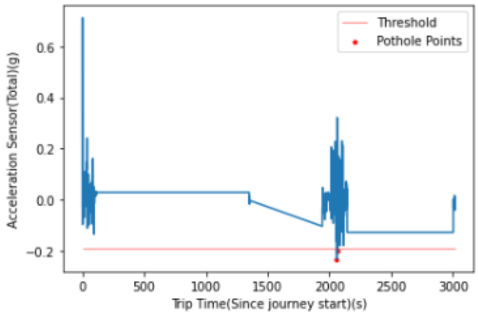
```
-------------------------------------------------------------------------

Scroll Up above the Map to View OUTPUT DATA and GRAPHS

Click on the icons on maps to see information of icon
-------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
```

**User Input ‘2’:**

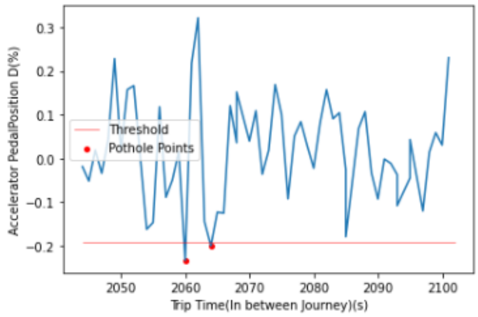
```
-------------------------------------------------------------------------

Scroll Up above the Map to View OUTPUT DATA and GRAPHS

Click on the icons on maps to see information of icon
-------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
2
```

Output (Part 1):

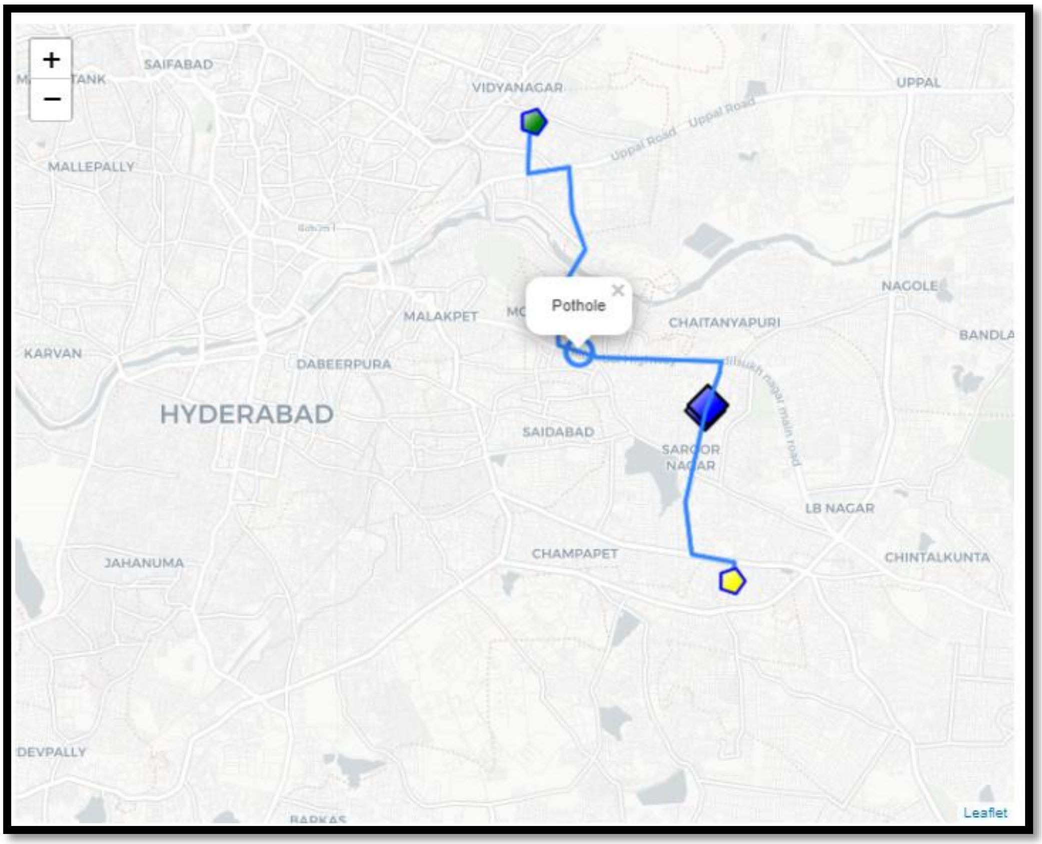
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
2
--------------------------------- Pothole Detected at --------------------------

 DEVICE TIME               LATITUDE                 LONGITUDE                ACCELEROMETER VALUE

00:58:35                  17.369277513333333       78.51759436666666        -0.23311444
00:58:39                  17.369317006666666       78.51745743333333        -0.19981611

----------------------------- 2 Pothole Detected  -------------------------------
```



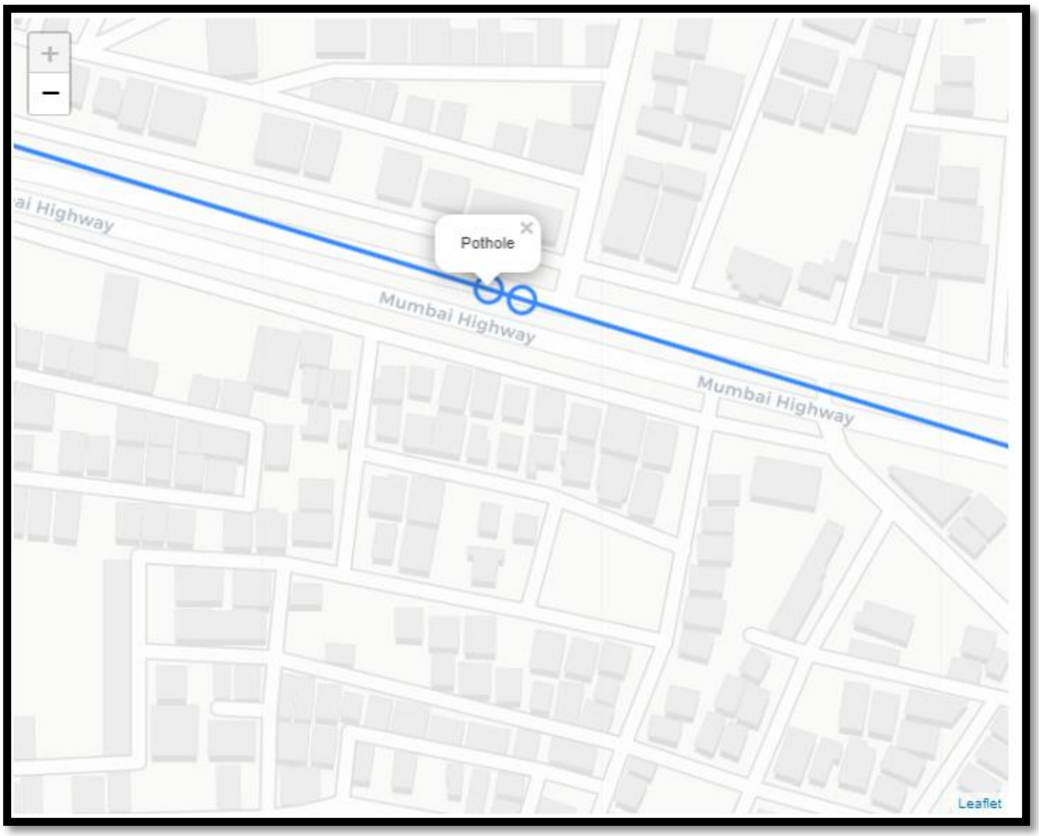

```
Zoom View of points
```




Output (Part 2):




Zoom View:




Output (Part 3):

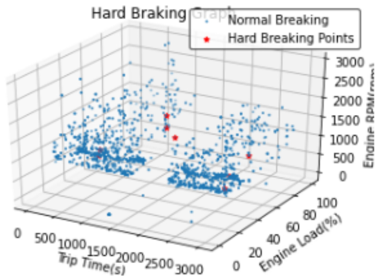
```
---------------------------------------------------------------------------

Scroll Up above the Map to View OUTPUT DATA and GRAPHS

Click on the icons on maps to see information of icon
---------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
```

**User Input '3':**

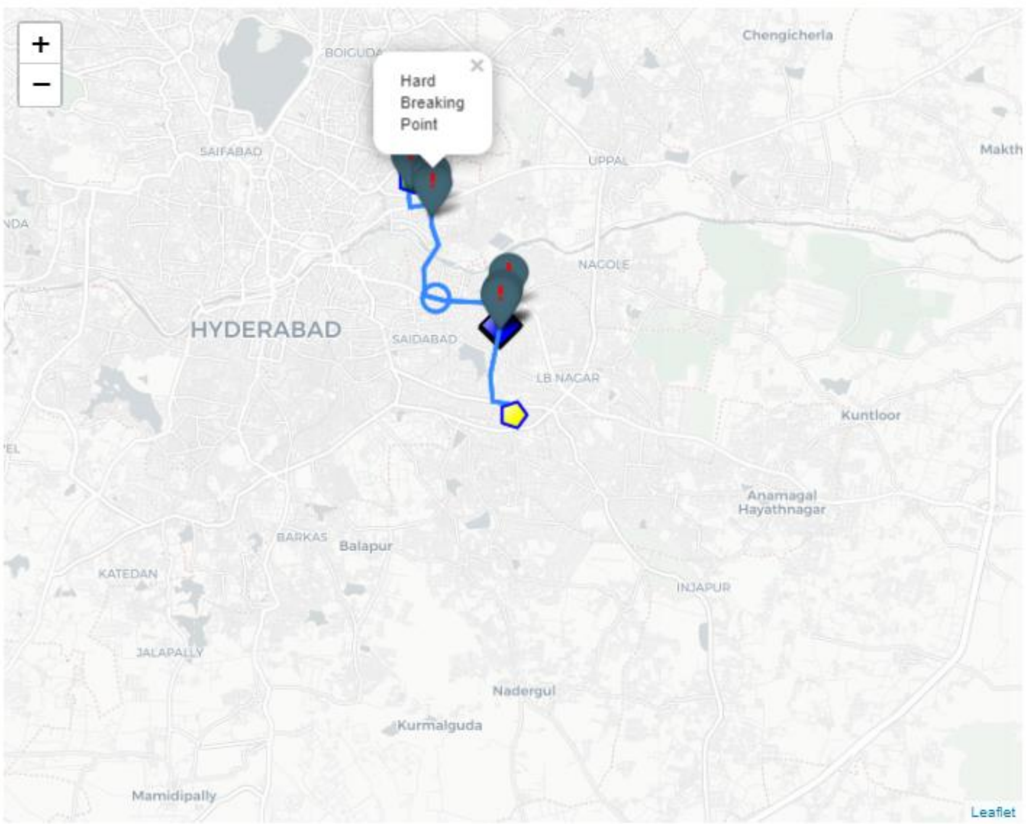
```
---------------------------------------------------------------------------

Scroll Up above the Map to View OUTPUT DATA and GRAPHS

Click on the icons on maps to see information of icon
---------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
3
```

Output Part 1:

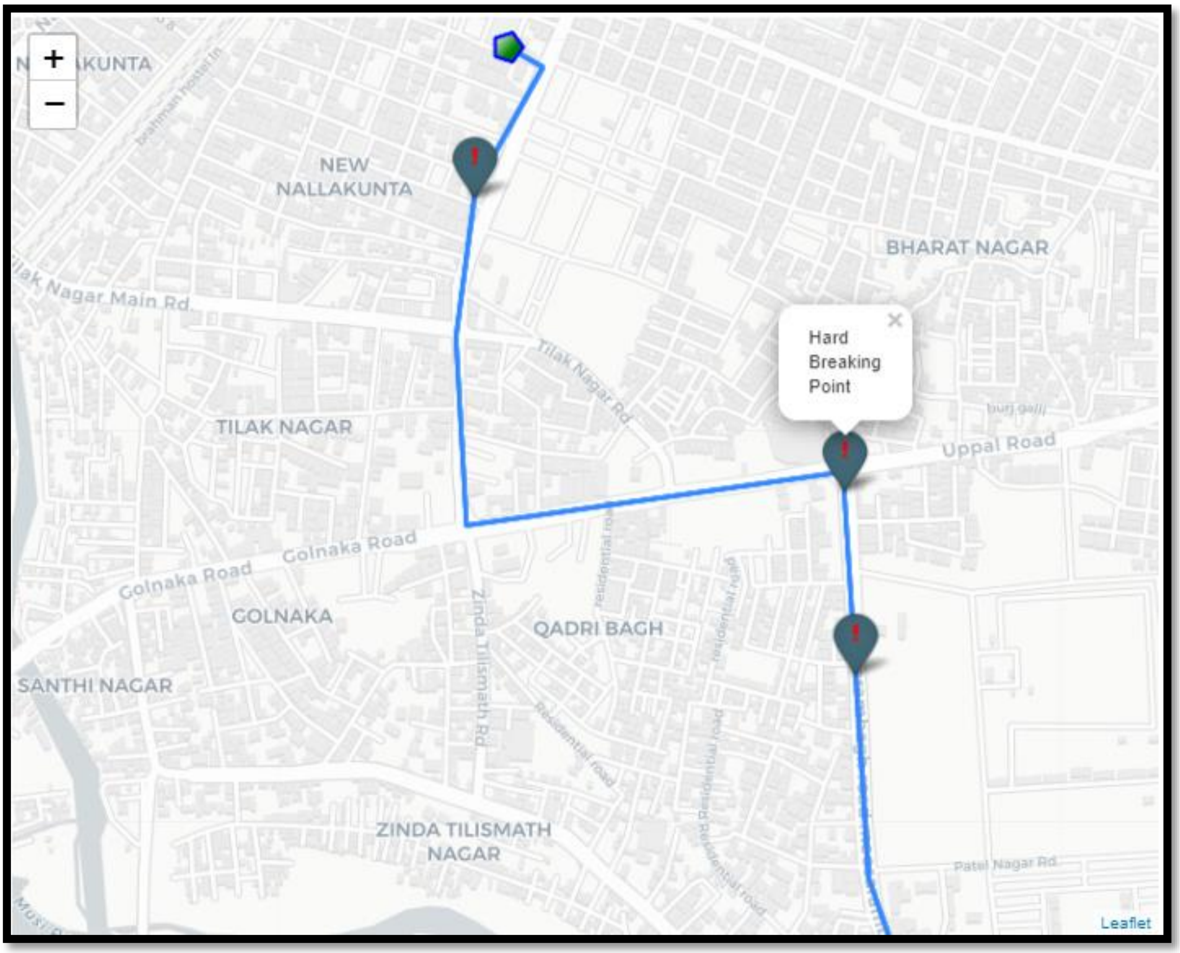
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
3
------------------------- Hard Braking Detected at -------------------------

LATITUDE                  LONGITUDE              ENGINE RPM DROPPED TO          ENGINE LOAD DROPPED TO

17.36221371111111         78.53380263703703      1110.5                         27.05882454
17.362827222222222        78.5340284074074       1068.5                         91.37255096
17.362933         78.53406733333333        983.0                24.31372643
17.363821533333333        78.53439431111111      799.0                  89.01960754
17.367659474452555        78.53570354014599      769.5                  78.4313736
17.38935138596491         78.51639728070175      1019.5                         60.39215851
17.39179357017544         78.51623309649122      849.5                  36.86274719
17.395730631578946        78.51105184210526      908.0                  18.8235302

-------------------- Hard Braking Detected 8 times -----------------------------
```




Output (Part 2):




Zoom View




Output (Part 3)

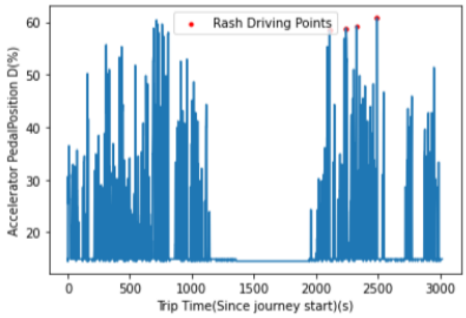
```
---------------------------------------------------------------------------

Scroll Up above the Map to View OUTPUT DATA and GRAPHS

Click on the icons on maps to see information of icon
---------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
```

**User Input ‘4’:**

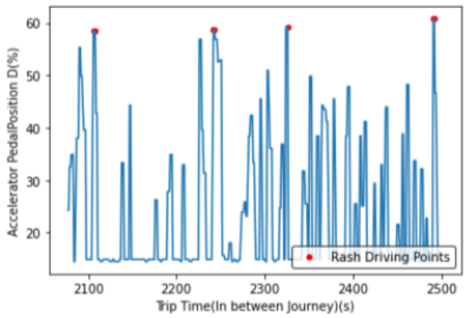
```
---------------------------------------------------------------------------

Scroll Up above the Map to View OUTPUT DATA and GRAPHS

Click on the icons on maps to see information of icon
---------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
4
```

Output (Part 1):

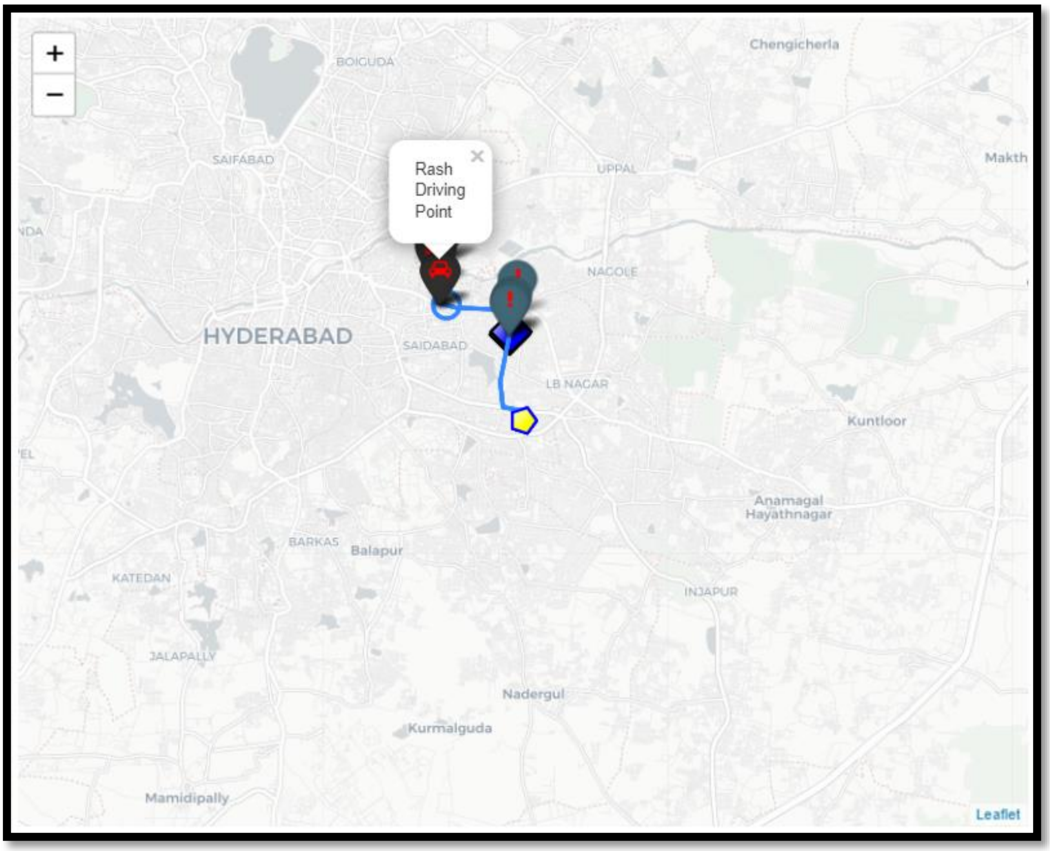
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
4
-------------------------- Rash Driving Detected at --------------------------

 DEVICE TIME               LATITUDE                  LONGITUDE                 PEDAL POSITION

00:59:21                  17.36973168666667         78.51601963333333        58.4313736
00:59:22                  17.36974156       78.51598539999999        58.4313736
00:59:23                  17.369751433333334        78.51595116666667        58.4313736
00:01:37                  17.37498105       78.51472439999999        58.8235321
00:01:38                  17.374990566666664        78.51472353333334        58.8235321
00:03:01                  17.37578045       78.5146516        59.2156868
00:05:46                  17.38070682442748         78.51746779389312        60.7843132
00:05:47                  17.38074924427481         78.51749593893129        60.7843132

--------------------- Rash Driving Detected 8 times ------------------------------
```



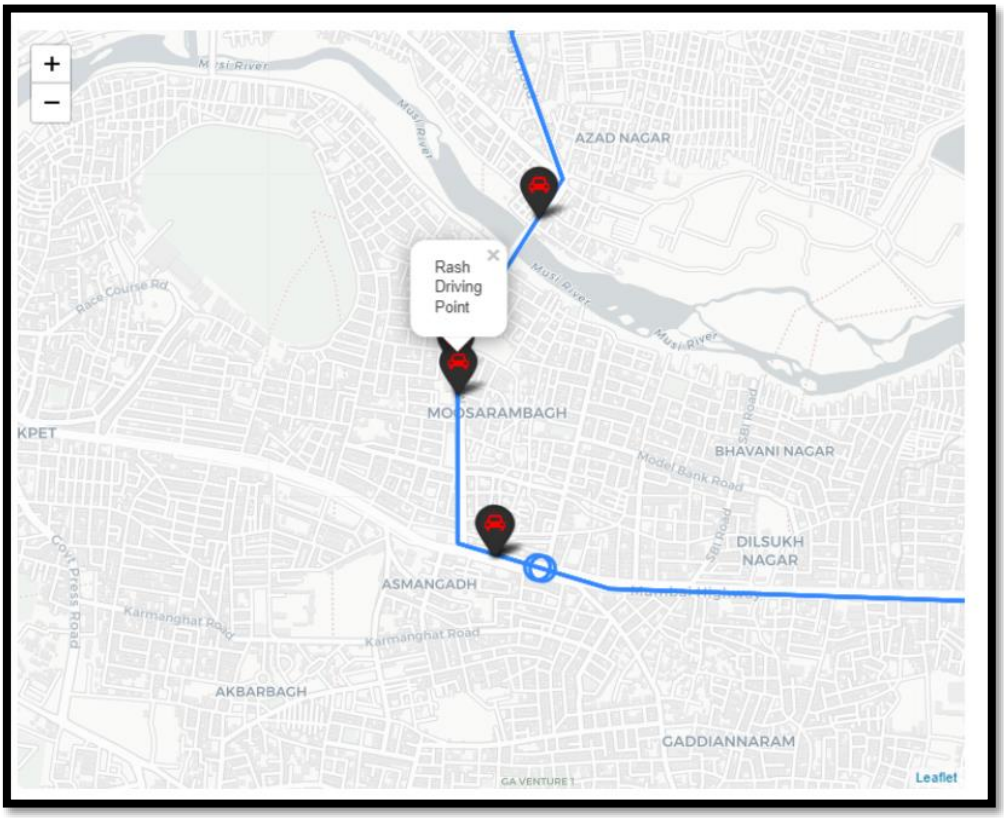

More Clear View of Rash Driving Points




Output (Part 2):



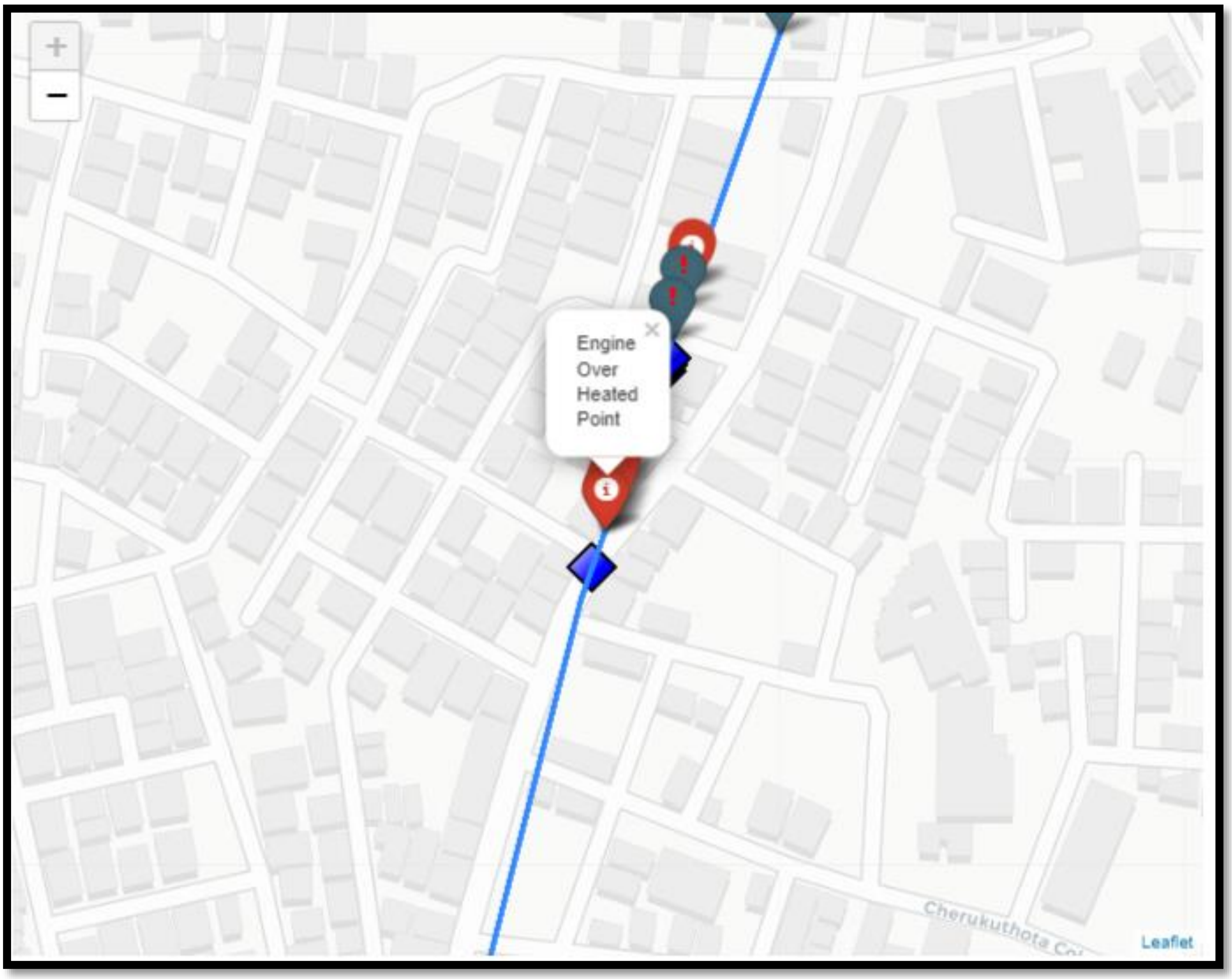

Zoom View:




**User Input '5':**

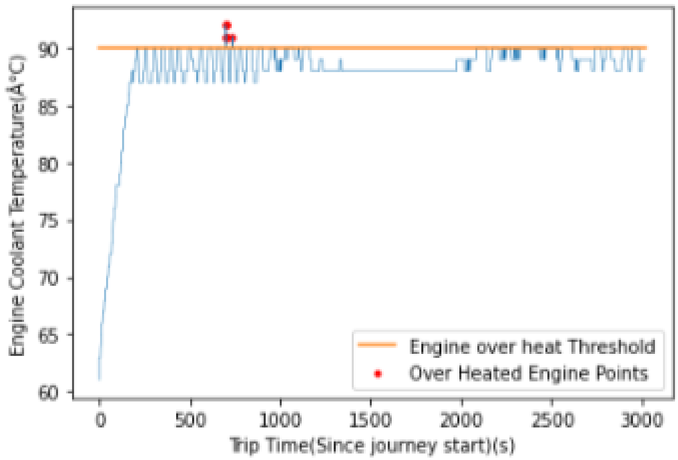
```
---------------------------------------------------------------------------
Scroll Up above the Map to View OUTPUT DATA and GRAPHS

Click on the icons on maps to see information of icon
---------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
5
```

Output (Part 2):

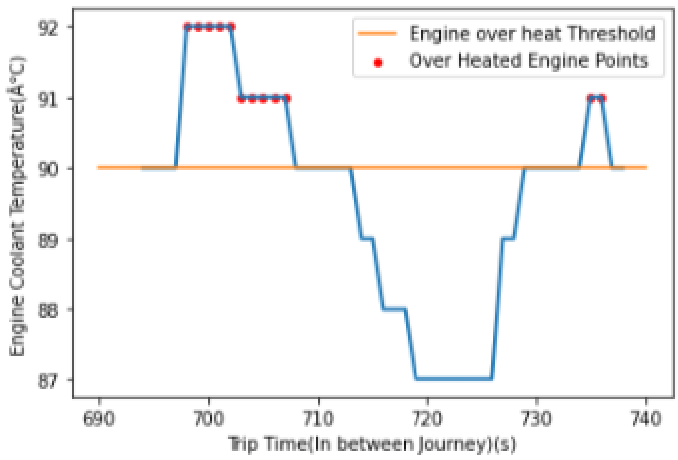


Output (Part 1):

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
5
----------------- Engine Over Heated at -----------------

 DEVICE TIME               LATITUDE         LONGITUDE        TEMPERATURE
00:35:53.600000          17.36221371111111        78.53380263703703        92
00:35:54.600000          17.362234866666668       78.53381042222222        92
00:35:55.600000          17.362256022222223       78.5338182074074         92
00:35:56.600000          17.362277177777777       78.53382599259258        92
00:35:57.600000          17.362298333333335       78.53383377777777        92
00:35:58.600000          17.36231948888889        78.53384156296296        91
00:35:59.600000          17.362340644444444       78.53384934814814        91
00:36:00.600000          17.3623618        78.53385713333333        91
00:36:01.600000          17.362382955555557       78.53386491851852        91
00:36:02.600000          17.36240411111111        78.5338727037037         91
00:36:30.600000          17.36299646666667        78.53409068888888        91
00:36:31.600000          17.363017622222223       78.53409847407407        91

--------------------- Over Heated 12 times ------------------------------
```




```
More Clear View of Over Heated Engine Points
```



Output (Part 3):

```
------------------------------------------------------------------------

Scroll Up above the Map to View OUTPUT DATA and GRAPHS

Click on the icons on maps to see information of icon
------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
```

**User Input ‘6’:**

One by one all the inferences will be displayed.

**User Input ‘7’**

```
----------------------------------------------------------------------------
Scroll Up above the Map to View OUTPUT DATA and GRAPHS

Click on the icons on maps to see information of icon
----------------------------------------------------------------------------

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
7
An exception has occurred, use %tb to see the full traceback.

SystemExit
```

## If user gives wrong input:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
8
Enter Correct Choice

Enter Number
-1
Enter Correct Choice

Enter Number
```

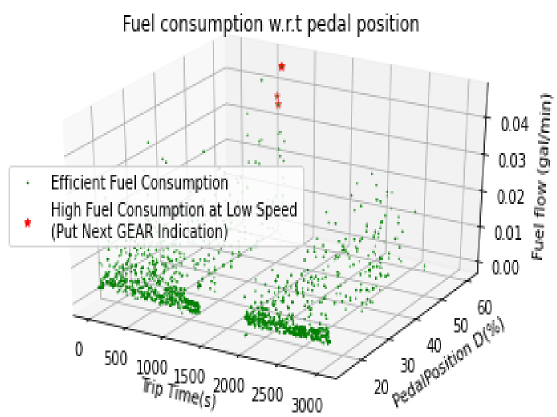
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Select Your Choice

1. High Fuel Consuption Detection (Put next gear or Go Slow)

2. Pothole Detection

3. Hard Braking

4. Rash Driving Detector

5. Engine Overheat Detection

6. Display All

7. Exit

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter Number
xyz
Wrong Input

Enter Number
```

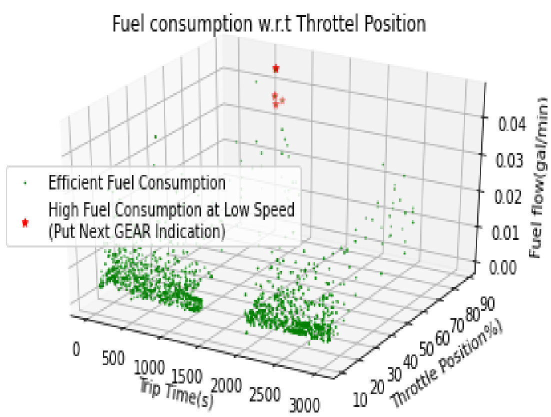
Exception Handling is achieved in program.

Video link for my output results (2:30 mins): https://drive.google.com/file/d/1yKCpnRGMqeSndWdJxy5Lz5s0vOgyrooA/view?usp=sharing

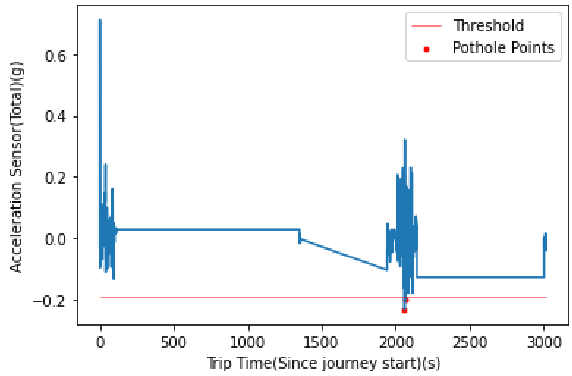
## 10. Graphs:

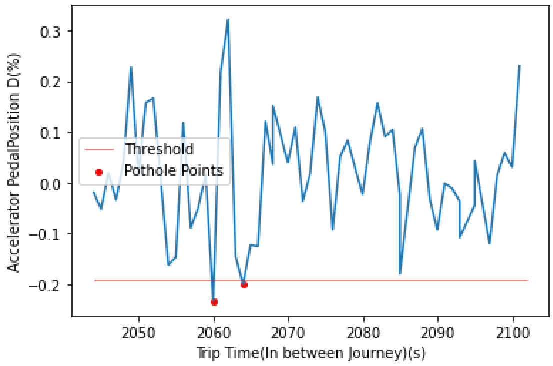
1. High Fuel Consumption Detection (Put next gear or Go Slow)



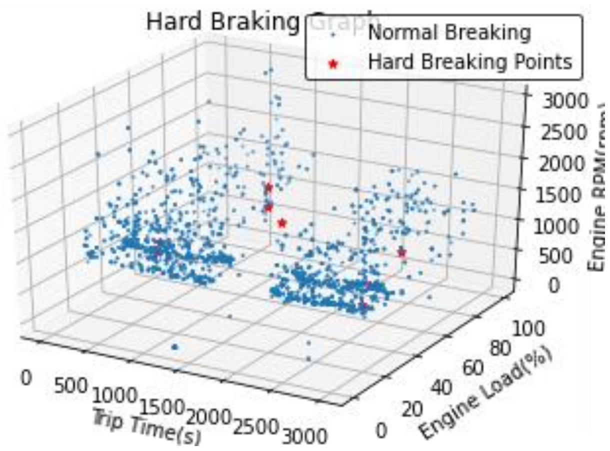




2. Pothole Detection

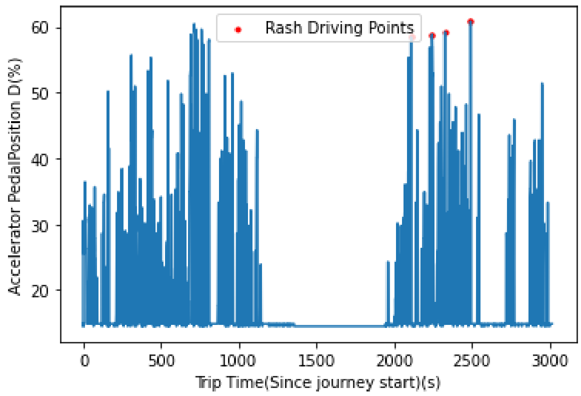


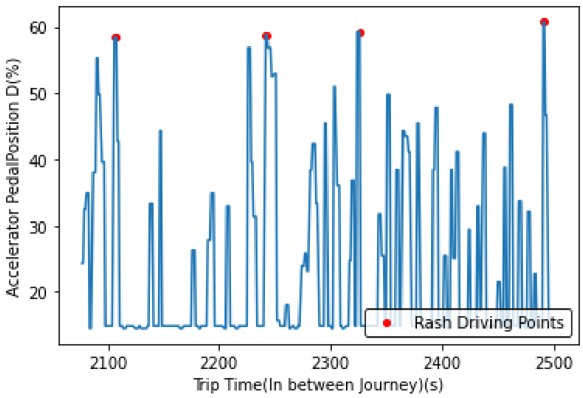




3. Hard Braking

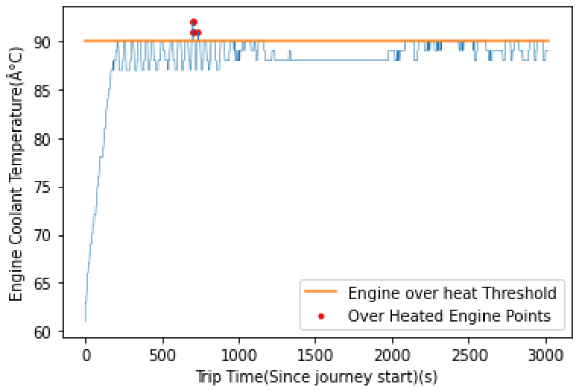


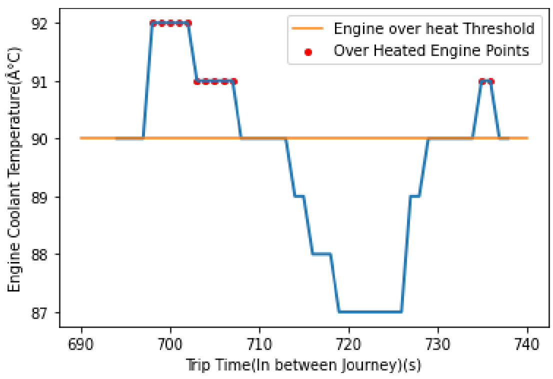

4. Rash Driving Detection



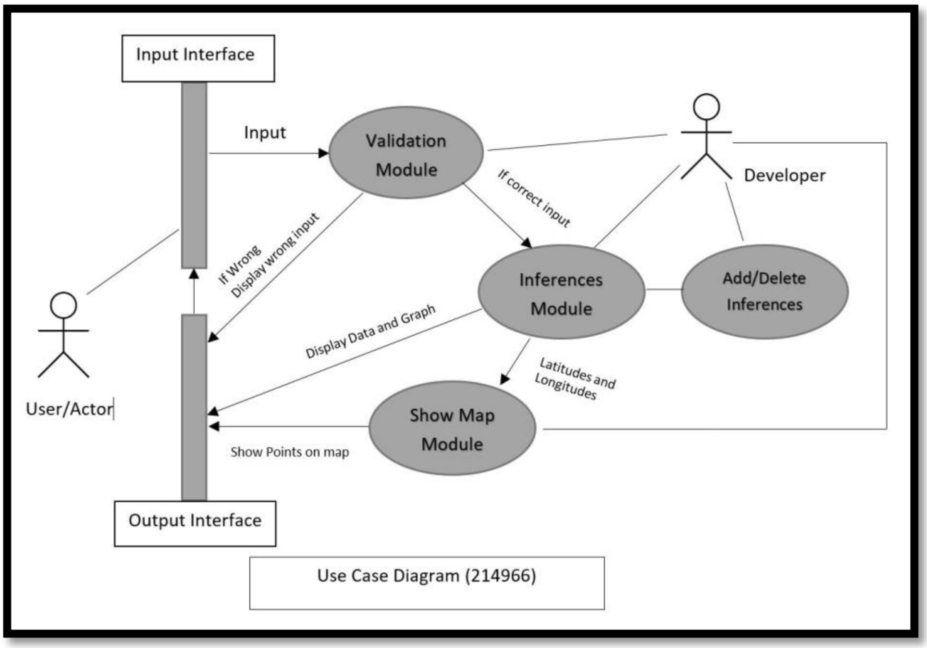

5. Engine Overheating Detection




------------------------------ End of Graph Section -------------------------------------

**Use Case Diagram:**




## 11. Code Snippet

Better view in Google Colab, Jupyter Notebook

```
#!/usr/bin/env python
# coding: utf-8

# # Created By:
#

# **Name:** Hrishikesh Shinde
#
# **Employee ID:** 214966
#
```

```
# **Email ID:** Hrishikesh.Shinde@kpit.com

# # Importing Libraries
# 


# 
# **Folium**: To plot Points on Map
# 
# *Installation: pip install folium*
# 
# ---
# 
# **Pandas**: To import dataset
# 
# *Installation: pip install pandas*
# 
# ---
# 
# **Matplotlib**: To plot Graphs
# 
# *Installation: python -m pip install -U matplotlib*
# 
# ---
# 
# **MPL_toolkits**: To plot 3 Dimensional Plot
# 
# *Installation: pip install --upgrade matplotlib*
# 
# ---
# 
# **Sys:** Used for exit
# 
# *Installation: Pre-Installed*

# In[1]:


import matplotlib.pyplot as plt
import pandas as pd
import folium
from mpl_toolkits.mplot3d import Axes3D
import sys


# # Importing Dataset

# OBD Dataset name should be data.xlsx
# 
```

```
# In[2]:


data = pd.read_excel('data.xlsx')


# # Function To Plot Points On Map and Displaying Map

# Here I have used Folium Library which helps us to plot points and route on m
ap.
#
# Also, different markers are used to display different inferences.
#
# **Click on the marker on map to view inference.**

# In[3]:


def show_map(latitude,longitude,mode):
  global data

  df = pd.DataFrame(list(zip(latitude, longitude)), columns =['Latitude', 'Lon
gitude'])
  folium.PolyLine([[17.34112692,78.53707726],[17.343478,78.537446],[17.344434,
78.532063],[17.350722, 78.531215],[17.354496,78.532169],[17.359157,78.532971],
[17.361981,78.533717],[17.364837, 78.534768],[17.366834, 78.535573],[17.368124
, 78.535777],[17.368616, 78.519888],[17.370097, 78.514753],[17.374667, 78.5147
53],[17.376380, 78.514597],[17.381937, 78.518284],[17.386574, 78.516584],[17.3
92033, 78.516217],[17.391306, 78.510949],[17.393834, 78.510797],[17.395836, 78
.511066],[17.397461, 78.512021],[17.39771674,78.51151644]]).add_to(map1)
  folium.RegularPolygonMarker(location=[17.34112692, 78.53707726],popup = 'Sta
rt Point',color ='blue',radius=10,fill_color='yellow',number_of_sides= 5).add_
to(map1)
  folium.RegularPolygonMarker(location=[17.39771674,78.51151644],popup = 'End
Point',color ='blue',radius=10,fill_color='green',number_of_sides= 5).add_to(m
ap1)

  #different marker to show different inferences
  if mode == 1:
    df.apply(lambda row:folium.CircleMarker(location=[row["Latitude"], row["Lo
ngitude"]],popup= 'Pothole').add_to(map1), axis=1)
  elif mode==2:
    df.apply(lambda row:folium.Marker(location  =[row["Latitude"], row["Longit
ude"]],popup = 'Engine Over Heated Point',icon=folium.Icon(color='red', icon =
'info-sign')).add_to(map1), axis=1)
  elif mode==3:
    df.apply(lambda row:folium.RegularPolygonMarker(location=[row["Latitude"],
 row["Longitude"]],popup = 'High Fuel Consumed Point \n(Put Next Gear or Go Sl
ow)').add_to(map1), axis=1)
```

```
  elif mode ==4:
    df.apply(lambda row:folium.Marker(location=[row["Latitude"], row["Longitud
e"]],popup = 'Rash Driving Point',icon =folium.Icon(color='black', icon='car',
 icon_color="red", prefix='fa')).add_to(map1), axis=1)
  elif mode == 5:
    df.apply(lambda row:folium.Marker(location=[row["Latitude"], row["Longitud
e"]],popup = 'Hard Breaking Point',icon =folium.Icon(color='cadetblue', icon='
exclamation', icon_color="red", prefix='fa')).add_to(map1), axis=1)
  display(map1)


# In[4]:


def map_display_all():
  next_gear()
  pothole_detection()
  hard_breaking()
  rash_driving()
  Engine_Overheat_Detection()


# # 1. Engine Overheat Detection

# Plot of time vs coolent temperature to check the over heating of engine
#
# OverHeating Threshold value for coolent temprature is set to 90°C
#
# **OverHeating Conditions:**
#
# 1) Coolent Temperature > above 90°C

# In[5]:


over_heat_latitude = []
over_heat_longitude = []
over_heat_time = []
over_heat_temp = []

def Engine_Overheat_Detection():
  global data
  global over_heat_latitude
  global over_heat_longitude
  global over_heat_time
  global over_heat_temp
  time = data['Trip Time(Since journey start)(s)']
  engine_temp = data['Engine Coolant Temperature(Â°C)']
  label = 'Over Heated Engine Points'
```

```
  over_heat_threshold_x = [[0],[data['Trip Time(Since journey start)(s)'][data
['Trip Time(Since journey start)(s)'].count()-1]]]
  over_heat_threshold_y = [[90],[90]]
  counter = 0
  plt.plot(time,engine_temp,linewidth =.5)
  plt.plot(over_heat_threshold_x,over_heat_threshold_y,label = 'Engine over he
at Threshold')

  print('----------------- Engine Over Heated at -----------------\n')
  print(' DEVICE TIME \t\t LATITUDE \t LONGITUDE \t TEMPERATURE')
  for n in range(data['Trip Time(Since journey start)(s)'].count()):
    if (data['Engine Coolant Temperature(Â°C)'][n]>90):
      over_heat_latitude.append(data[' Latitude'][n])
      over_heat_longitude.append(data[' Longitude'][n])
      over_heat_time.append(data['Trip Time(Since journey start)(s)'][n])
      over_heat_temp.append(data['Engine Coolant Temperature(Â°C)'][n])
      counter +=1
      print(f"{data[' Device Time'][n]} \t {data[' Latitude'][n]} \t {data[' L
ongitude'][n]} \t {data['Engine Coolant Temperature(Â°C)'][n]}")
  print(f'\n--------------------- Over Heated {counter} times ----------------
---------------\n')
  plt.scatter(over_heat_time,over_heat_temp,label = label,color='red',marker='
.')
  label = "_nolegend_"
  plt.legend()
  plt.xlabel('Trip Time(Since journey start)(s)')
  plt.ylabel('Engine Coolant Temperature(Â°C)')
  plt.show()
  print('\n\n')
  Engine_Overheat_Detection_zoom()


# Zoom View to get a clear picture of overheated points

# In[6]:


def Engine_Overheat_Detection_zoom():
  global data
  global over_heat_latitude
  global over_heat_longitude
  global over_heat_time
  global over_heat_temp

  time = data['Trip Time(Since journey start)(s)'][705:750]
  engine_temp = data['Engine Coolant Temperature(Â°C)'][705:750]
  over_heat_threshold_x = [[690],[740]]
  over_heat_threshold_y = [[90],[90]]
  label = 'Over Heated Engine Points'
```

```
  plt.plot(time,engine_temp,linewidth =2)
  print('\n\nMore Clear View of Over Heated Engine Points')
  plt.plot(over_heat_threshold_x,over_heat_threshold_y,label = 'Engine over he
at Threshold')
  plt.scatter(over_heat_time,over_heat_temp,label = label,color='red',marker='
.',linewidths=2)
  label = "_nolegend_"
  plt.legend()
  plt.xlabel('Trip Time(In between Journey)(s)')
  plt.ylabel('Engine Coolant Temperature(Â°C)')
  plt.show()
  print('\n\n')

  show_map(over_heat_latitude,over_heat_longitude,2)


# # 2. Rash Driving Detection

# Plotted points when driver was rash driving
#
# Used Pedal Position sensor threshold value = 58
#
# Also checked weather the car is in forward motion or not, using acceleromete
r X-axis values
#
# ---
#
#
#
# **Rash Driving Conditions**:
#
# When Car is moving accelerometer's
#
#   1) Positive values indicate an increase in velocity.
#
#   2) Negative values indicate an decrease in velocity.
#
#   3) Zero values indicate constant velocity
#
# Here we considered the positive value and zero value to confirm the vehicle
is motion or not.
#
# Then we detected the value if pedal position.
#
# Accelerator PedalPosition D(%)>58
#
# Engine RPM > 800
#
```

```
#
# If both condition match rash driving is detected
#

# In[7]:


rash_latitude = []
rash_longitude = []
rash_time = []
rash_pedal_position = []

def rash_driving():
  global data
  global rash_latitude
  global rash_longitude
  global rash_time
  global rash_pedal_position

  time = data['Trip Time(Since journey start)(s)']
  pedal_position = data['Accelerator PedalPosition D(%)']

  plt.plot(time,pedal_position)
  counter = 0
  print('-------------------------- Rash Driving Detected at -----------------
---------\n')
  print(' DEVICE TIME \t\t LATITUDE \t\t LONGITUDE \t\t PEDAL POSITION \n')
  for n in range(data['Accelerator PedalPosition D(%)'].count()):
    if data['Accelerator PedalPosition D(%)'][n]>58 and data['Acceleration Sen
sor(X axis)(g)'][n]>=0 and data['Engine RPM(rpm)'][n]>900:
      counter += 1
      rash_latitude.append(data[' Latitude'][n])
      rash_longitude.append(data[' Longitude'][n])
      rash_time.append(data['Trip Time(Since journey start)(s)'][n])
      rash_pedal_position.append(data['Accelerator PedalPosition D(%)'][n])
      print(f"{data[' Device Time'][n]} \t\t {data[' Latitude'][n]} \t {data['
 Longitude'][n]} \t {data['Accelerator PedalPosition D(%)'][n]}")

  print(f'\n---------------------- Rash Driving Detected {counter} times ------
-------------------------')
  plt.scatter(rash_time,rash_pedal_position,label ='Rash Driving Points',color
='red',marker='.')
  plt.legend()
  plt.xlabel('Trip Time(Since journey start)(s)')
  plt.ylabel('Accelerator PedalPosition D(%)')
  plt.show()

  rash_driving_zoom_view()
```

```
# Zoom View to get a clear picture of Rash Driving points

# In[8]:


def rash_driving_zoom_view():
  global data
  global rash_latitude
  global rash_longitude
  global rash_time
  global rash_pedal_position

  time = data['Trip Time(Since journey start)(s)'][1503:1928]
  pedal_position = data['Accelerator PedalPosition D(%)'][1503:1928]
  print('\n\nMore Clear View of Rash Driving Points')
  plt.plot(time,pedal_position)
  plt.scatter(rash_time,rash_pedal_position,label = 'Rash Driving Points',colo
r='red',marker='.',linewidths=2)
  label = "_nolegend_"
  plt.legend(loc = 'lower right',edgecolor= 'black')
  plt.xlabel('Trip Time(In between Journey)(s)')
  plt.ylabel('Accelerator PedalPosition D(%)')
  plt.show()
  print('\n\n')
  show_map(rash_latitude,rash_longitude,4)


# # 3. Pothole Detection

# Function To detect the number of potholes
#
# **Pothole Detection Conditions:**
#
#   1) Threshold for Acceleration Total (g) <-0.19
#

# In[9]:


pothole_latitude = []
pothole_longitude = []
pothole_time = []
pothole_value = []

def pothole_detection():
  global data
  global pothole_latitude
```

```
  global pothole_longitude
  global pothole_time
  global pothole_value

  time = data['Trip Time(Since journey start)(s)']
  accleration = data['Acceleration Sensor(Total)(g)']

  thresholdx = [[0],[data['Trip Time(Since journey start)(s)'][data['Trip Time
(Since journey start)(s)'].count()-1]]]
  thresholdy = [[-0.19],[-0.19]]

  plt.plot(time,accleration)
  plt.plot(thresholdx,thresholdy,linewidth = 0.5, color = 'red',label = 'Thres
hold')
  plt.plot()
  pothole_counter = 0
  print('--------------------------------- Pothole Detected at ---------------
-----------\n')
  print(' DEVICE TIME \t\t LATITUDE \t\t LONGITUDE \t\t ACCELEROMETER VALUE\n'
)
  for n in range(data['Trip Time(Since journey start)(s)'].count()):
    if(data['Acceleration Sensor(Total)(g)'][n]<-.19):
      pothole_counter +=1
      pothole_latitude.append(data[' Latitude'][n])
      pothole_longitude.append(data[' Longitude'][n])
      pothole_time.append(data['Trip Time(Since journey start)(s)'][n])
      pothole_value.append(data['Acceleration Sensor(Total)(g)'][n])
      print(f"{data[' Device Time'][n]} \t\t {data[' Latitude'][n]} \t {data['
 Longitude'][n]} \t {data['Acceleration Sensor(Total)(g)'][n]}")
  print(f'\n-----------------------------
- {pothole_counter} Pothole Detected  -------------------------------\n')
  plt.scatter(pothole_time,pothole_value,label ='Pothole Points',color='red',m
arker='.')
  plt.legend()
  plt.xlabel('Trip Time(Since journey start)(s)')
  plt.ylabel('Acceleration Sensor(Total)(g)')
  plt.show()
  print('\n\nZoom View of points\n')
  pothole_detection_zoom_view()


# Zoom view of pothole points to get a better picture of it

# In[10]:


def pothole_detection_zoom_view():
  global data
  global pothole_latitude
```

```python
  global pothole_longitude
  global pothole_time
  global pothole_value
  time = data['Trip Time(Since journey start)(s)'][1470:1528]
  pedal_position = data['Acceleration Sensor(Total)(g)'][1470:1528]

  thresholdx = [[data['Trip Time(Since journey start)(s)'][1470]],[data['Trip Time(Since journey start)(s)'][1528]]]
  thresholdy = [[-0.19],[-0.19]]

  plt.plot(time,pedal_position)
  plt.plot(thresholdx,thresholdy,linewidth = 0.5, color = 'red',label = 'Threshold')
  plt.scatter(pothole_time,pothole_value,label ='Pothole Points',color='red',marker='.',linewidths=2)
  label = "_nolegend_"
  plt.legend(loc ='center left')
  plt.xlabel('Trip Time(In between Journey)(s)')
  plt.ylabel('Accelerator PedalPosition D(%)')
  plt.show()
  print('\n\n')
  show_map(pothole_latitude,pothole_longitude,1)


# # 4. High Fuel Consuption Detection (Put Next Gear or Go Slow)

# To check high consuption fuel points.
#
# When driver tries to run his vehicle on same gear without putting into next gear. High fuel consuption is detected.
#
# **Alerts Driver to Put next Gear or Go Slow**
#
# **Conditions To Detect:**
#
# 1) Acceleration Pedal Positon > 55
#
# 2) Fuel Consuption Rate > 0.029
#
# 3) Throttel Position > 80
#

# In[11]:


extra_fuel_consuption_latitude = []
extra_fuel_consuption_longitude = []
extra_fuel_consuption_time = []
extra_fuel_consuption_value = []
```

```
extra_fuel_paddle_position = []
extra_fuel_throttle_position = []

def next_gear():
  global data
  global extra_fuel_consuption_latitude
  global extra_fuel_consuption_longitude
  global extra_fuel_consuption_time
  global extra_fuel_consuption_value
  global extra_fuel_paddle_position
  global extra_fuel_throttle_position

  fig = plt.figure(figsize=plt.figaspect(0.25))
  ax = fig.add_subplot(121, projection='3d')

  high_fuel_consuption_counter =0
  time =data['Trip Time(Since journey start)(s)']
  pedal_position =data['Accelerator PedalPosition D(%)']
  fuel =data['Fuel flow rate/minute(gal/min)']
  throttel = data['Throttle Position(Manifold)(%)']

  print('------------------------ INDICATION TO PUT NEXT GEAR or GO SLOW -----
-----------------\n')
  print('-------------------------- High Fuel Consuption Detected at ---------
-----------------\n')
  print(' DEVICE TIME \t\t LATITUDE \t\t LONGITUDE \t\t HIGH FUEL CONSUPTION\n
')
  for n in range(data['Trip Time(Since journey start)(s)'].count()):
    if (data['Accelerator PedalPosition D(%)'][n] >55 and data['Fuel flow rate
/minute(gal/min)'][n]>0.029 and data['Throttle Position(Manifold)(%)'][n]>80):
      high_fuel_consuption_counter +=1
      extra_fuel_consuption_latitude.append(data[' Latitude'][n])
      extra_fuel_consuption_longitude.append(data[' Longitude'][n])
      extra_fuel_consuption_time.append(data['Trip Time(Since journey start)(s
)'][n])
      extra_fuel_consuption_value.append(data['Fuel flow rate/minute(gal/min)'
][n])
      extra_fuel_paddle_position.append(data['Accelerator PedalPosition D(%)']
[n])
      extra_fuel_throttle_position.append(data['Throttle Position(Manifold)(%)
'][n])
      print(f"{data[' Device Time'][n]} \t {data[' Latitude'][n]} \t {data[' L
ongitude'][n]} \t {data['Fuel flow rate/minute(gal/min)'][n]}")

  print(f'\n--------------------
- High Fuel Consuption Detected {high_fuel_consuption_counter} times ---------
----------------------\n')
```

```
  ax.scatter(time, pedal_position, fuel, c='g', marker='o',s=0.3,label = 'Effi
cient Fuel Consumption')
  ax.scatter(extra_fuel_consuption_time,extra_fuel_paddle_position,extra_fuel_
consuption_value,c='r',marker='*',label = 'High Fuel Consumption at Low Speed\
n(Put Next GEAR Indication)')
  ax.set_title('Fuel consumption w.r.t pedal position')
  ax.legend(loc = 'center left')
  ax.set_xlabel('Trip Time(s)')
  ax.set_ylabel('PedalPosition D(%)')
  ax.set_zlabel('Fuel flow (gal/min)')


  ax = fig.add_subplot(122, projection='3d')
  ax.scatter(time, throttel, fuel, c='g', marker='o',s=0.3,label = 'Efficient
Fuel Consumption')
  ax.scatter(extra_fuel_consuption_time,extra_fuel_throttle_position,extra_fue
l_consuption_value,c='r',marker='*',label = 'High Fuel Consumption at Low Spee
d\n(Put Next GEAR Indication)')
  ax.set_title('Fuel consumption w.r.t Throttel Position')
  ax.legend(loc ='center left')
  ax.set_xlabel('Trip Time(s)')
  ax.set_ylabel('Throttle Position%)')
  ax.set_zlabel('Fuel flow(gal/min)')

  plt.show()
  print('\n\n')
  show_map(extra_fuel_consuption_latitude,extra_fuel_consuption_longitude,3)


# # 5. Hard Braking Detection

# Hard Breaking Points detected.
#
# When there is sudden drop in engine rpm and engine load hard braking is dete
cted
#
# **Conditions to detect:**
#
# 1) Difference between two consecutive Engine RPM should be more than -950
#
# 2) Difference between two consecutive Engine Load should be more than -70

# In[12]:


hard_breaking_latitude = []
hard_breaking_longitude = []
```

```
def hard_breaking():
  global data
  global hard_breaking_latitude
  global hard_breaking_longitude

  time = data['Trip Time(Since journey start)(s)']
  load = data['Engine Load(%)']
  rpm = data['Engine RPM(rpm)']
  hard_breaking_counter = 0
  ax = plt.axes(projection='3d')
  ax.scatter(time, load, rpm,s=1,label = 'Normal Breaking')
  hard_breaking_label = 'Hard Breaking Points'
  print('-------------------------- Hard Braking Detected at ------------------
---------\n')
  print('LATITUDE \t\t LONGITUDE \t\t ENGINE RPM DROPPED TO\t\t ENGINE LOAD DR
OPPED TO\n')
  for n in range(data['Trip Time(Since journey start)(s)'].count()-1):
    if data['Engine RPM(rpm)'][n+1] - data['Engine RPM(rpm)'][n]<-
950 and data['Engine Load(%)'][n+1] - data['Engine Load(%)'][n]>-70:
      hard_breaking_counter +=1
      hard_breaking_latitude.append(data[' Latitude'][n+1])
      hard_breaking_longitude.append(data[' Longitude'][n+1])
      ax.scatter(data['Trip Time(Since journey start)(s)'][n+1],data['Engine L
oad(%)'][n+1],data['Engine RPM(rpm)'][n+1],color = 'r',linewidth =1,marker = '
*',label = hard_breaking_label)
      hard_breaking_label = '__nolabel__'
      print(f"{data[' Latitude'][n+1]} \t {data[' Longitude'][n+1]} \t {data['
Engine RPM(rpm)'][n+1]} \t\t\t {data['Engine Load(%)'][n+1]}")
  print(f'\n--------------------
- Hard Braking Detected {hard_breaking_counter} times -----------------------
-------\n')
  ax.legend(edgecolor= 'black')
  ax.set_title('Hard Braking Graph')
  ax.set_xlabel('Trip Time(s)')
  ax.set_ylabel('Engine Load(%)')
  ax.set_zlabel('Engine RPM(rpm)')
  plt.show()
  print('\n\n')
  show_map(hard_breaking_latitude,hard_breaking_longitude,5)


# # User Choice
#

# As per the user's choice the inference will be displayed
#

# In[13]:
```

```
def user_choice(choice):
  if choice == 1:
    next_gear()
    print_last()
    main()
  elif choice == 2:
    pothole_detection()
    print_last()
    main()
  elif choice == 3:
    hard_breaking()
    print_last()
    main()
  elif choice == 4:
    rash_driving()
    print_last()
    main()
  elif choice == 5:
    Engine_Overheat_Detection()
    print_last()
    main()
  elif choice == 6:
    map_display_all()
    print_last()
    main()
  elif choice == 7:
    sys.exit()


# In[14]:


def print_last():
  print('---------------------------------------------------------------------
-----\n')
  print('Scroll Up above the Map to View OUTPUT DATA and GRAPHS\n')
  print('Click on the icons on maps to see information of icon')
  print('---------------------------------------------------------------------
-----\n')


# # Program Starts Here (Main)
#
#


# In[15]:


def main():
```

```
  print('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Inferences ~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~')
  print('\nSelect Your Choice')
  print('\n1. High Fuel Consuption Detection (Put next gear or Go Slow)\n\n2.
Pothole Detection\n\n3. Hard Braking\n\n4. Rash Driving Detector\n\n5. Engine
Overheat Detection\n\n6. Display All\n\n7. Exit\n')
  print('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~')
  while(1):
    try:
      choice = int(input('Enter Number\n'))
      if choice>=8 or choice<=0:
        print('Enter Correct Choice\n')
        continue
      else:
        break
    except:
      print('Wrong Input\n')
  user_choice(choice)


if __name__ == "__main__":
  map1 = folium.Map(location=[17.34112692,78.53707726],tiles='cartodbpositron'
,zoom_start=12,width=750, height=600)
  main()


# In[ ]:
```

------------------------------------------ END OF THE CODE -------------
------------------------------

-----------------------------------------------THANK YOU----------------
--------------------------------