ENTS 749C: ADVANCED TOPICS IN NETWORKING - VEHICULAR NETWORKS

MINI PROJECT #1

Submitted by

PRATHAPANI, NIKHIL

UID: 113-66-0344

EMAIL: nprathap@umd.edu

Section: 0101

A. JAMES CLARK
SCHOOL OF ENGINEERING

LEARN MORE, GO FURTHER.
It's That Simple!

TABLE OF CONTENTS

Title Page……………………………………………………………………………………………………0

Problem Statement………………………………………………………………………………………..2

I. Assumptions……………………………………………………………………………………..2
II. Python based Data Analysis……………………………………………………………………..2
III. Data Plots…………………………………………………………………………………………14
IV. Observations……………………………………………………………………………….....21

PROBLEM STATEMENT:
Alice drives her Controller Area Network (CAN) enabled car. She has volunteered to make her vehicle's CAN bus data available to an insurance company XYZ for the purpose of earning the "safe driver" price reduction. Her vehicle data can be used to create position traces and monitor some of vehicular system parameters when moving on these traces.
XYZ obtains the data file called alicedata.json. Now XYZ has to mine this data file to assess the scenario that Alice's vehicle went through and learn her driving behavior. Python data must be processed and the desired plots must be generated.

I. Assumptions:

There are 12 different signal names in the given data.
The following parameters are assumed to have following units:

1. Vehicle Speed (KM/HR) –kilometer/hour
2. Accelerator Pedal Position (Radians)
3. Engine Speed (RPM-rotations per minute)
4. Torque at transmission (lb-feet)
5. Latitude (degrees)
6. Longitude (degrees)
7. Steering Wheel Angle (degrees)
8. Fuel consumed since restart (gallon)
9. Odometer (KM)
10. Fuel Level (Gallon)
11. Brake Pedal Status (True/False)
12. Transmission gear position (neutral/first/second/third/fourth)

The given alicedata.json file is loaded as a list of dictionaries and suitable operations have been performed on it.

Each dictionary has the keys: 'Name', 'Time Stamp', 'Value' and each of them have corresponding values.

II. Python-based Data Analysis:

- func1()

Code:

```
import json
data = []
def func1(alicedata):
    with open(alicedata) as json_file:
        for line in json_file:
            global data
            data.append(json.loads(line)) #reading data from the json file
            #print(line)
        return data #returning the function for calling it later

#e1=func1('alicedata.json')
```

Output:

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import prathapani_proj1 as pp
>>> a=pp.func1('alicedata.json')
>>> 
```

- func2()

Code:

```
def func2(alicedata):
    e1=func1(alicedata) #retrieving the output list of function 1 , by calling it along with the parameter.
    from pprint import pprint
    #pprint(e1[:10])
    for ten_entries_data_file in e1[:10]:
        pprint(ten_entries_data_file)
#func2('alicedata.json')
```

Output:

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import prathapani_proj1 as pp
>>> pp.func2('alicedata.json')
{'name': 'vehicle_speed', 'timestamp': 1364310855.004, 'value': 0}
{'name': 'accelerator_pedal_position', 'timestamp': 1364310855.004, 'value': 0}
{'name': 'engine_speed', 'timestamp': 1364310855.005, 'value': 0}
{'name': 'torque_at_transmission', 'timestamp': 1364310855.008, 'value': -3}
{'name': 'vehicle_speed', 'timestamp': 1364310855.008, 'value': 0}
{'name': 'accelerator_pedal_position', 'timestamp': 1364310855.008, 'value': 0}
{'name': 'engine_speed', 'timestamp': 1364310855.013, 'value': 0}
{'name': 'latitude', 'timestamp': 1364310855.013, 'value': 40.797997}
{'name': 'longitude', 'timestamp': 1364310855.013, 'value': -73.964027}
{'name': 'torque_at_transmission', 'timestamp': 1364310855.017, 'value': -3}
>>> 
```

- func3()

Code:

```
def func3(alicedata):
    e1=func1(alicedata)
    lst_signal_names_unique=[]
    lst1_signal_names_all=[]
    lst_1=[]
    lst_a=[]
    count_dict_number_of_occurences = {}
```

```
for signal_name_unique in e1:
    if not signal_name_unique['name'] in lst_signal_names_unique: #ignoring repititions
        lst_signal_names_unique.append(signal_name_unique['name'])
print(lst_signal_names_unique)

for signal_name_all in e1:
    lst1_signal_names_all.append(signal_name_all['name'])

for word in lst1_signal_names_all:
    if word in count_dict_number_of_occurences: #finding number of occurences
        count_dict_number_of_occurences[word]+= 1
    else:
        count_dict_number_of_occurences[word] = 1
print(count_dict_number_of_occurences)

'''
#This is another way of finding the number of occurences of a list element
from collections import Counter
c=Counter(lst1)
print(c)
'''

inp=input("Enter a SIGNAL NAME from the following: -->
'vehicle_speed','accelerator_pedal_position', 'engine_speed', 'torque_at_transmission', 'latitude',
'longitude', 'steering_wheel_angle', 'fuel_consumed_since_restart', 'odometer','fuel_level',
'brake_pedal_status', 'transmission_gear_position'\n")
if(inp!='transmission_gear_position'):
    for i in e1:
        lst1_signal_names_all.append(i['name'])
        if(i['name']== inp):
            lst_1.append(i['value'])
    print("The minimum value of",inp, "is" ,min(lst_1),"\n","The maximum value
of",inp,"is",max(lst_1))
    print("number of occurrences and value range of ",inp,"
is:",count_dict_number_of_occurences[inp])
elif(inp=='transmission_gear_position'): #transmission_gear_position has string values. Hence
mapping it to integers for future use
    for i in e1:
        if(i['name']== 'transmission_gear_position'):
            lst_a.append(i['value'])
    #print(lst_a)
    for k in range(len(lst_a)):
        if(lst_a[k]=='first'):
            lst_a[k]=1
        elif(lst_a[k]=='second'):
            lst_a[k]=2
```

```
            elif(lst_a[k]=='three'):
                lst_a[k]=3
            elif(lst_a[k]=='fourth'):
                lst_a[k]=4
            else:
                lst_a[k]=0 #for 'neutral' gear
        #print(lst_a)
        print("The minimum value of",inp, "is" ,min(lst_a),"\n","The maximum value of",inp,"is",max(lst_a))
        print("number of occurrences and value range of ",inp," is:",len(lst_a))

#func3('alicedata.json')
```

Output:

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import prathapani_proj1 as pp
>>> pp.func3('alicedata.json')
['vehicle_speed', 'accelerator_pedal_position', 'engine_speed', 'torque_at_trans
mission', 'latitude', 'longitude', 'steering_wheel_angle', 'fuel_consumed_since_
restart', 'odometer', 'fuel_level', 'brake_pedal_status', 'transmission_gear_pos
ition']
{'vehicle_speed': 29442, 'brake_pedal_status': 35, 'torque_at_transmission': 294
01, 'longitude': 297, 'engine_speed': 29388, 'latitude': 297, 'accelerator_pedal
_position': 29414, 'steering_wheel_angle': 990, 'fuel_level': 2971, 'odometer':
2970, 'fuel_consumed_since_restart': 2970, 'transmission_gear_position': 26}
Enter a SIGNAL NAME from the following: --> 'vehicle_speed','accelerator_pedal_p
osition', 'engine_speed', 'torque_at_transmission', 'latitude', 'longitude', 'st
eering_wheel_angle', 'fuel_consumed_since_restart', 'odometer','fuel_level', 'br
ake_pedal_status', 'transmission_gear_position'
vehicle_speed
The minimum value of vehicle_speed is 0
 The maximum value of vehicle_speed is 47.68
number of occurrences and value range of  vehicle_speed  is: 29442
>>> |
```

- func4()

Code:

```
def func4(alicedata):
    e1=func1(alicedata)
    lst_odometer_reading=[]
    lst1_trip_time=[]


    for odometer_reading in e1:
        if(odometer_reading['name']== 'odometer'):
            lst_odometer_reading.append(odometer_reading['value'])
```

```
print("The vehicle trip distance over recorded data:", lst_odometer_reading[-1]-lst_odometer_reading[0],"kilometer or",0.621371*(lst_odometer_reading[-1]-lst_odometer_reading[0]),"miles")
#print("The vehicle trip distance over recorded data:", max(lst)-min(lst),"kilometer or",0.621371*(max(lst)-min(lst)),"miles")

for trip_time in e1:
    if(trip_time['name']== 'vehicle_speed'):
        lst1_trip_time.append(trip_time['timestamp'])
print("total trip time period:",lst1_trip_time[-1]-lst1_trip_time[0],"seconds")

'''
#This is another way of finding distance between two points using spherical coordinates. Latitude and Longitude in this case
inp='latitude'
inp1='longitude'

for i in e1:
    if(i['name']== 'latitude'):
        lst_1.append(i['value'])
print("1st and last latitude of recorded data:",lst_1[-1],lst_1[0])
lat1=lst_1[0]
lat2=lst_1[-1]

for i in e1:
    if(i['name']== 'longitude'):
        lst_2.append(i['value'])
print("1st and last longitude of recorded data:",lst_2[-1],lst_2[0])
lon1=lst_2[0]
lon2=lst_2[-1]

from math import radians, cos, sin, asin, sqrt
#def distancelatlon(lon1, lat1, lon2, lat2):

"""
Calculate the great circle distance between two points
on the earth (specified in decimal degrees)
"""

    # convert decimal degrees to radians
lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    #latitude-longitude formula
dlon = lon2 - lon1
dlat = lat2 - lat1
a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
c = 2 * asin(sqrt(a))
```

```
    miles = 3956.2691* c
    #print("The vehicle trip distance over recorded data:", miles," miles")
    '''
#func4('alicedata.json')
```

Output:

```
>>> pp.func4('alicedata.json')
The vehicle trip distance over recorded data: 1.8847660000010364 kilometer or 1.
171138934186644 miles
total trip time period: 296.9909999370575 seconds
>>> |
```

- func5()

Code:

```
def func5(alicedata):
    e1=func1(alicedata)

    lst1=[]
    lst2=[]
    lst3=[]
    lst4=[]
    lst5=[]
    lst6=[]
    lst7=[]
    lst8=[]
    lst9=[]
    lst10=[]
    lst11=[]
    lst12=[]
    lst_1=[]
    lst_a=[]

    from matplotlib import pylab as pl
    from matplotlib import pyplot as pl
    fig = pl.figure()
    import numpy as np

    for time_stamp in e1:
        lst_1.append(time_stamp['timestamp'])

    for i in e1:
        if(i['name']== 'vehicle_speed'):
            lst1.append(i['value'])
    pl.plot(lst_1[0:len(lst1)],lst1)
    fig.text(0.8,0.8,'vehicle speed vs time stamp',ha='center')
```

```
pl.xlabel('timestamp')
pl.ylabel('vehicle_Speed')
pl.show()
#fig.suptitle('vehicle speed vs trip time', fontsize=20)
#plt.xlabel('trip time', fontsize=18)
#plt.ylabel('vehicle speed', fontsize=18)

fig = pl.figure()
for i in e1:
    if(i['name']== 'accelerator_pedal_position'):
        lst2.append(i['value'])
pl.plot(lst_1[0:len(lst2)],lst2)
fig.text(0.8,0.8,'accelerator pedal position vs time stamp',ha='center')
pl.xlabel('timestamp')
pl.ylabel('accelerator_pedal_position')
pl.show()

fig = pl.figure()
for i in e1:
    if(i['name']== 'engine_speed'):
        lst3.append(i['value'])
pl.plot(lst_1[0:len(lst3)],lst3)
fig.text(0.8,0.8,'engine speed vs time stamp',ha='center')
pl.xlabel('timestamp')
pl.ylabel('engine_speed')
pl.show()

fig = pl.figure()
for i in e1:
    if(i['name']== 'torque_at_transmission'):
        lst4.append(i['value'])
pl.plot(lst_1[0:len(lst4)],lst4)
fig.text(0.8,0.8,'torque_at_transmission vs time stamp',ha='center')
pl.xlabel('timestamp')
pl.ylabel('torque_at_transmission')
pl.show()

fig = pl.figure()
for i in e1:
    if(i['name']== 'latitude'):
        lst5.append(i['value'])
pl.plot(lst_1[0:len(lst5)],lst5)
fig.text(0.8,0.8,'latitude vs time stamp',ha='center')
pl.xlabel('timestamp')
pl.ylabel('latitude')
pl.show()
```

```
fig = pl.figure()
for i in e1:
    if(i['name']== 'longitude'):
        lst6.append(i['value'])
pl.plot(lst_1[0:len(lst6)],lst6)
fig.text(0.8,0.8,'longitude vs time stamp',ha='center')
pl.xlabel('timestamp')
pl.ylabel('longitude')
pl.show()

fig = pl.figure()
for i in e1:
    if(i['name']== 'steering_wheel_angle'):
        lst7.append(i['value'])
pl.plot(lst_1[0:len(lst7)],lst7)
fig.text(0.8,0.8,'steering_wheel_angle vs time stamp',ha='center')
pl.xlabel('timestamp')
pl.ylabel('steering_wheel_angle')
pl.show()

fig = pl.figure()
for i in e1:
    if(i['name']== 'fuel_consumed_since_restart'):
        lst8.append(i['value'])
pl.plot(lst_1[0:len(lst8)],lst8)
fig.text(0.8,0.8,'fuel_consumed_since_restart vs time stamp',ha='center')
pl.xlabel('timestamp')
pl.ylabel('fuel_consumed_since_restart')
pl.show()

fig = pl.figure()
for i in e1:
    if(i['name']== 'odometer'):
        lst9.append(i['value'])
pl.plot(lst_1[0:len(lst9)],lst9)
fig.text(0.8,0.8,'odometer vs time stamp',ha='center')
pl.xlabel('timestamp')
pl.ylabel('odometer')
pl.show()

fig = pl.figure()
for i in e1:
    if(i['name']== 'fuel_level'):
        lst10.append(i['value'])
pl.plot(lst_1[0:len(lst10)],lst10)
```

```
fig.text(0.8,0.8,'fuel_level vs time stamp',ha='center')
pl.xlabel('timestamp')
pl.ylabel('fuel_level')
pl.show()

fig = pl.figure()
for i in e1:
    if(i['name']== 'brake_pedal_status'):
        lst11.append(i['value'])
pl.plot(lst_1[0:len(lst11)],lst11)
fig.text(0.8,0.8,'brake_pedal_status vs time stamp',ha='center')
pl.xlabel('timestamp')
pl.ylabel('brake_pedal_status')
pl.show()

fig = pl.figure()
for i in e1:
        if(i['name']== 'transmission_gear_position'):
            lst_a.append(i['value'])
    #print(lst_a)
for k in range(len(lst_a)):
    if(lst_a[k]=='first'):
        lst_a[k]=1
    elif(lst_a[k]=='second'):
        lst_a[k]=2
    elif(lst_a[k]=='three'):
        lst_a[k]=3
    elif(lst_a[k]=='fourth'):
        lst_a[k]=4
    else:
        lst_a[k]=0 #for 'neutral' gear
pl.plot(lst_1[0:len(lst_a)],lst_a)
fig.text(0.8,0.8,'transmission_gear_position',ha='center')
pl.xlabel('timestamp')
pl.ylabel('transmission_gear_position')
pl.show()


#func5('alicedata.json')
```

Output: *The outputs are plotted in dataplots section of report

```
>>> pp.func5('alicedata.json')
>>> |
```

- func6()

Code:

```
def func6(alicedata):
    e1=func1(alicedata)
    inp='vehicle_speed'
    lst_signal_name=[]
    lst1_vehicle_speed_values=[]
    lst_1=[]
    count = {}
    for i in e1:
        if(i['name']== 'vehicle_speed'):
            lst1_vehicle_speed_values.append(i['value'])

    for i in e1:
        lst_signal_name.append(i['name'])

    for word in lst_signal_name:
        if word in count:
            count[word]+= 1
        else:
            count[word] = 1


    print("The minimum value of vehicle_speed is" ,min(lst1_vehicle_speed_values),"\n","The maximum value of vehicle_speed is",max(lst1_vehicle_speed_values))
    print("number of occurrences of vehicle_speed is:",count[inp])
    print("Average speed of Alice's vehicle:",sum(lst1_vehicle_speed_values)/count[inp],"kilometer/hour")
    print("Maximum speed of Alice's vehicle:",max(lst1_vehicle_speed_values),"kilometer/hour")

#func6('alicedata.json')
```

Output:

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import prathapani_proj1 as pp
>>> pp.func6('alicedata.json')
The minimum value of vehicle_speed is 0
 The maximum value of vehicle_speed is 47.68
number of occurrences of vehicle_speed is: 29442
Average speed of Alice's vehicle: 22.9733691624547 kilometer/hour
Maximum speed of Alice's vehicle: 47.68 kilometer/hour
>>>
```

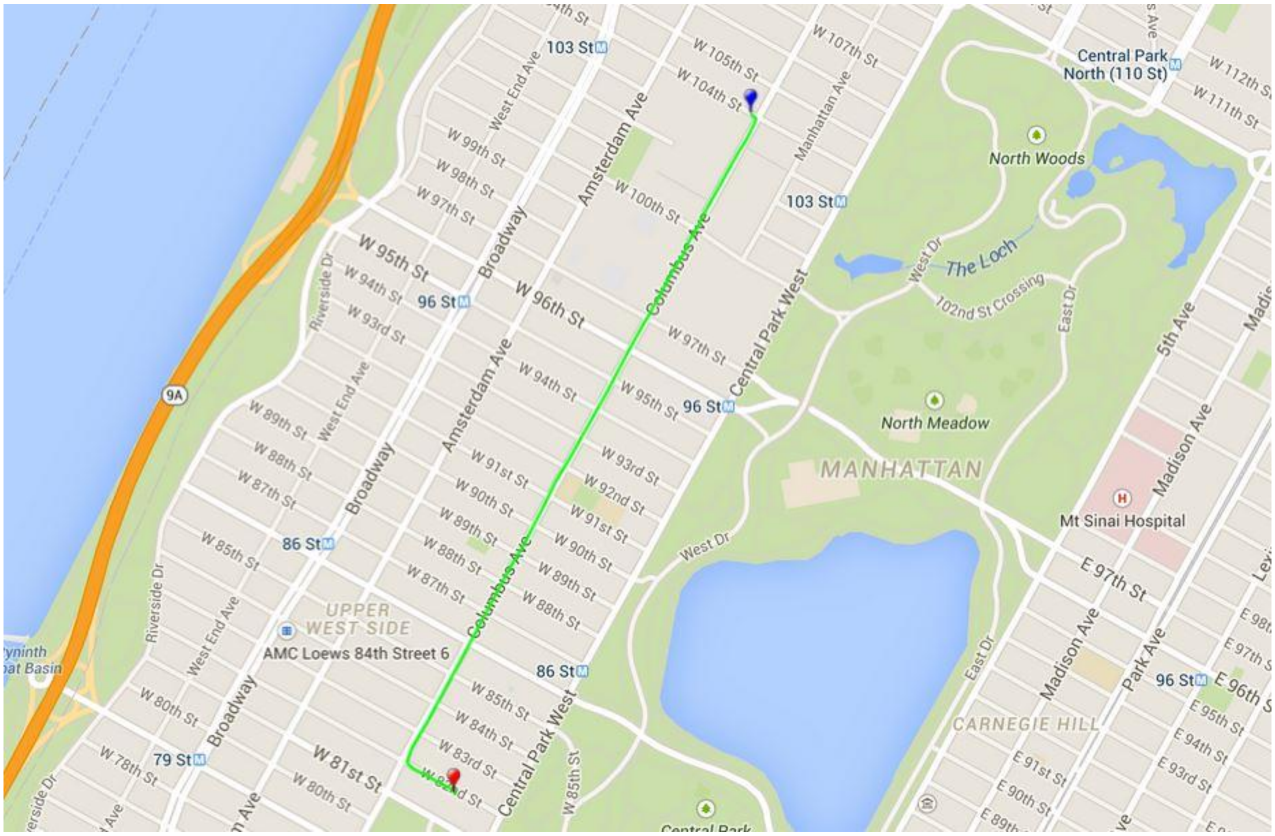
- func7()

Code:

```
def func7(alicedata):
    e1=func1(alicedata)
    lst1_latitude=[]
    lst2_longitude=[]
    import pygmaps
    #import webbrowser

    for latitude_values in e1:
        if(latitude_values['name']== 'latitude'):
            lst1_latitude.append(latitude_values['value']) #appending all latitude values to a list

    for longitude_values in e1:
        if(longitude_values['name']== 'longitude'):
            lst2_longitude.append(longitude_values['value']) #appending all longitude values to a list

    mymap = pygmaps.maps(lst1_latitude[int((len(lst1_latitude))/2)],
lst2_longitude[int((len(lst2_longitude))/2)], 16)
    path = list(zip(lst1_latitude,lst2_longitude)) #creating a new list with
    mymap.addpath(path,"#00FF00")
    mymap.addpoint(lst1_latitude[0], lst2_longitude[0], "#0000FF") #starting point - Blue
    mymap.addpoint(lst1_latitude[-1], lst2_longitude[-1], "#FF0000") #ending point - Red
    mymap.draw('./mymap.html')

#func7('alicedata.json')
```

Output:

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import prathapani_proj1 as pp
>>> pp.func7('alicedata.json')
>>> 
```



- func8()

Code:

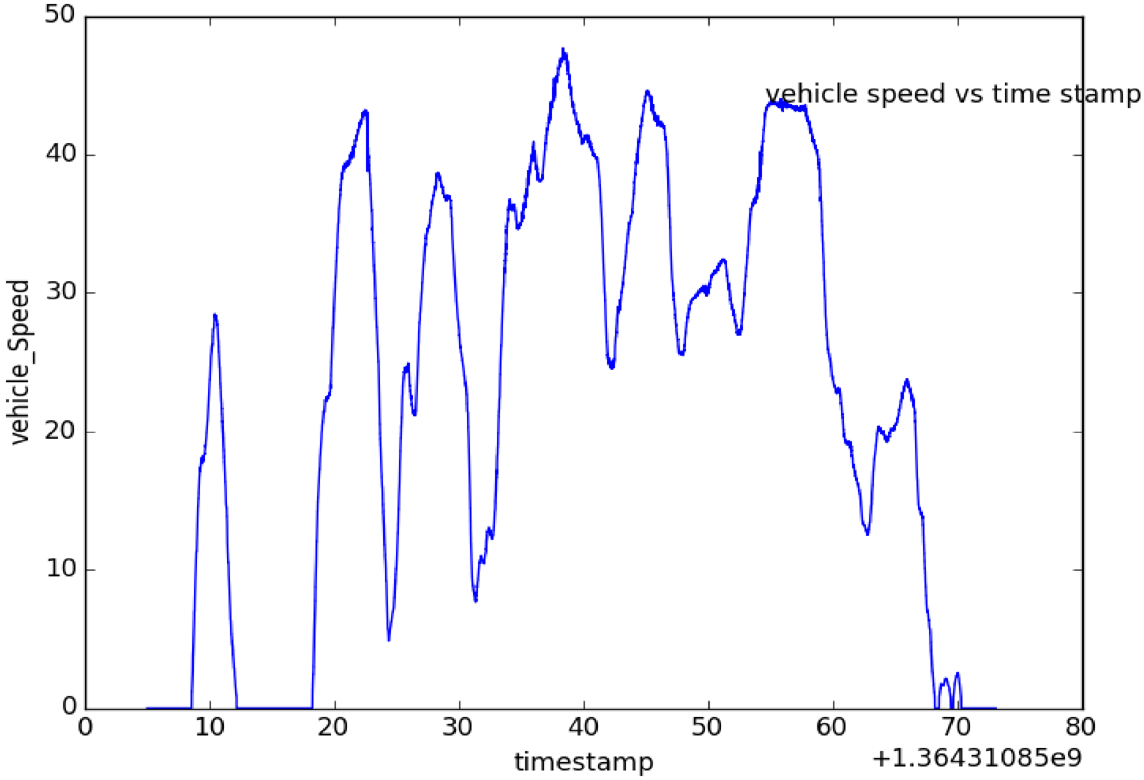
```
def func8(alicedata):
    e1=func1(alicedata)
    lst_odometer_reading=[]
    lst1_fuel_consumed=[]


    for odometer_reading in e1:
        if(odometer_reading['name']== 'odometer'):
            lst_odometer_reading.append(odometer_reading['value'])
    print("The vehicle trip distance over recorded data:", lst_odometer_reading[-1]-
lst_odometer_reading[0],"kilometer or",0.621371*(lst_odometer_reading[-1]-
lst_odometer_reading[0]),"miles")

    for fuel_level in e1:
        if(fuel_level['name']== 'fuel_consumed_since_restart'):
            lst1_fuel_consumed.append(fuel_level['value'])
    print("total fuel consumed for the trip:",lst1_fuel_consumed[-1]-lst1_fuel_consumed[0],"Gallons")

    print("The gas Mileage of Alice's Vehicle is:", (0.621371*(lst_odometer_reading[-1]-
lst_odometer_reading[0]))/(lst1_fuel_consumed[-1]-lst1_fuel_consumed[0]),"miles per gallon")

#func8('alicedata.json')
```

Output:

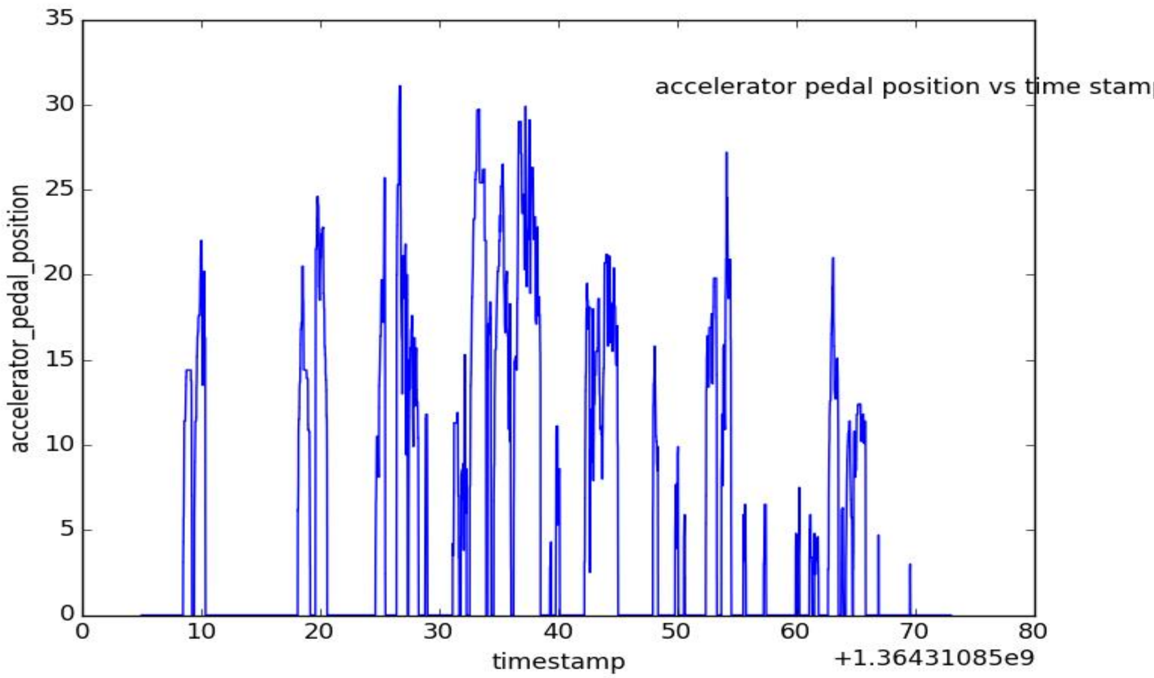
```
>>> pp.func8('alicedata.json')
The vehicle trip distance over recorded data: 1.8847660000010364 kilometer or 1.
171138934186644 miles
total fuel consumed for the trip: 0.19486400000000004 Gallons
The gas Mileage of Alice's Vehicle is: 6.010032300407689 miles per gallon
>>> 
```

III. Data Plots:

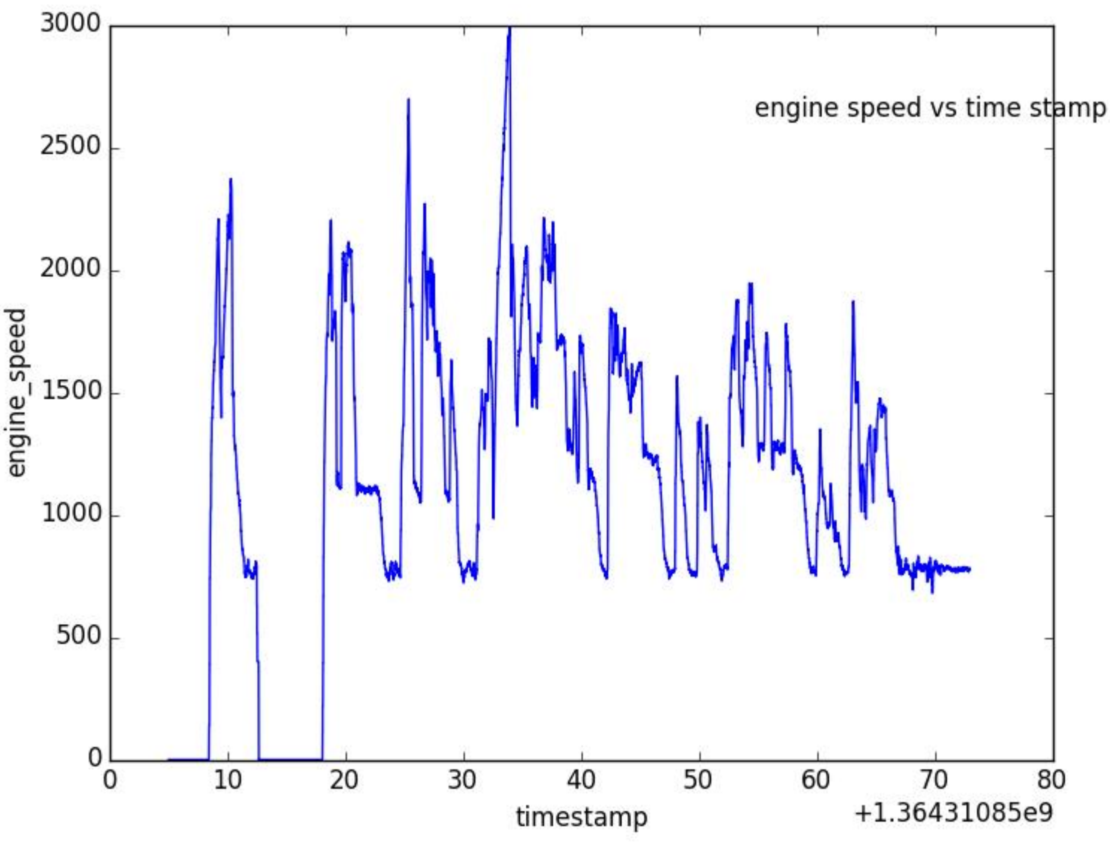
func5():
III-a) vehicle speed vs time stamp



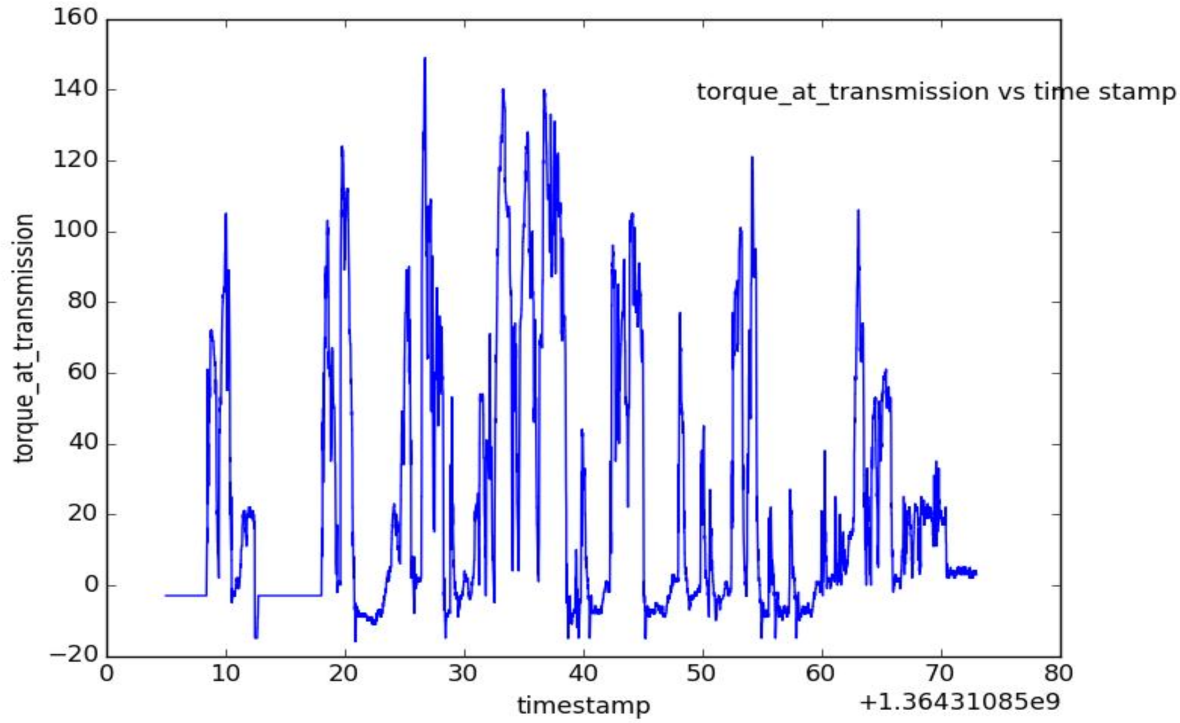
III-b) accelerator pedal position vs time stamp



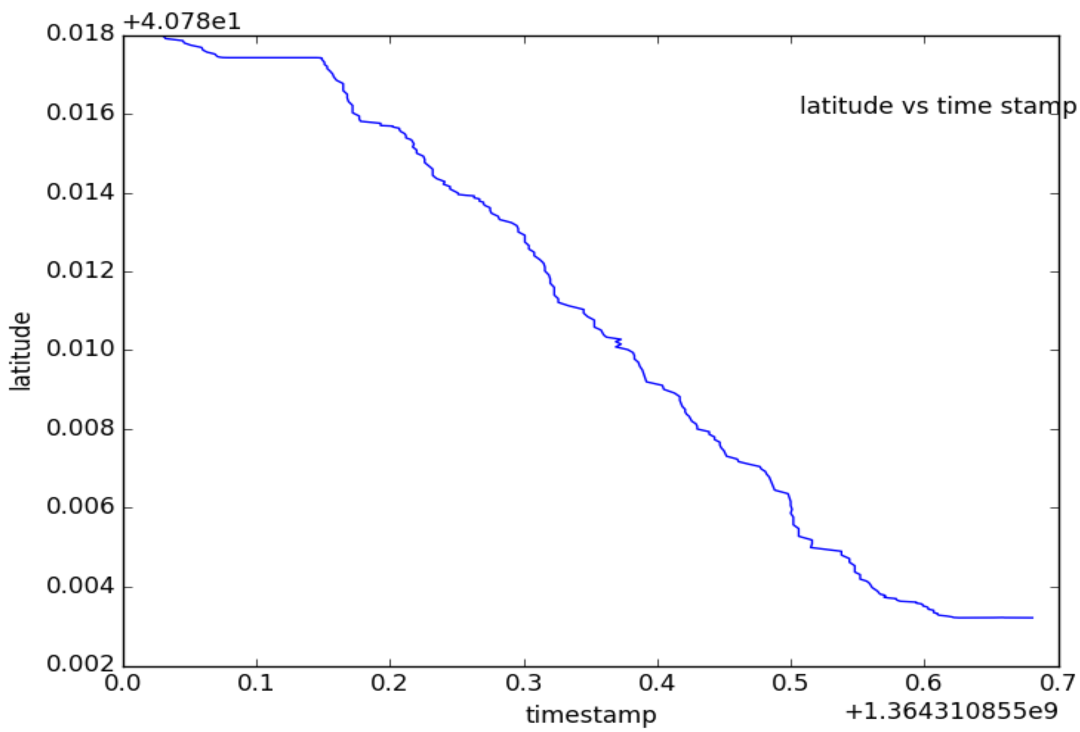
III-c) engine speed vs time stamp



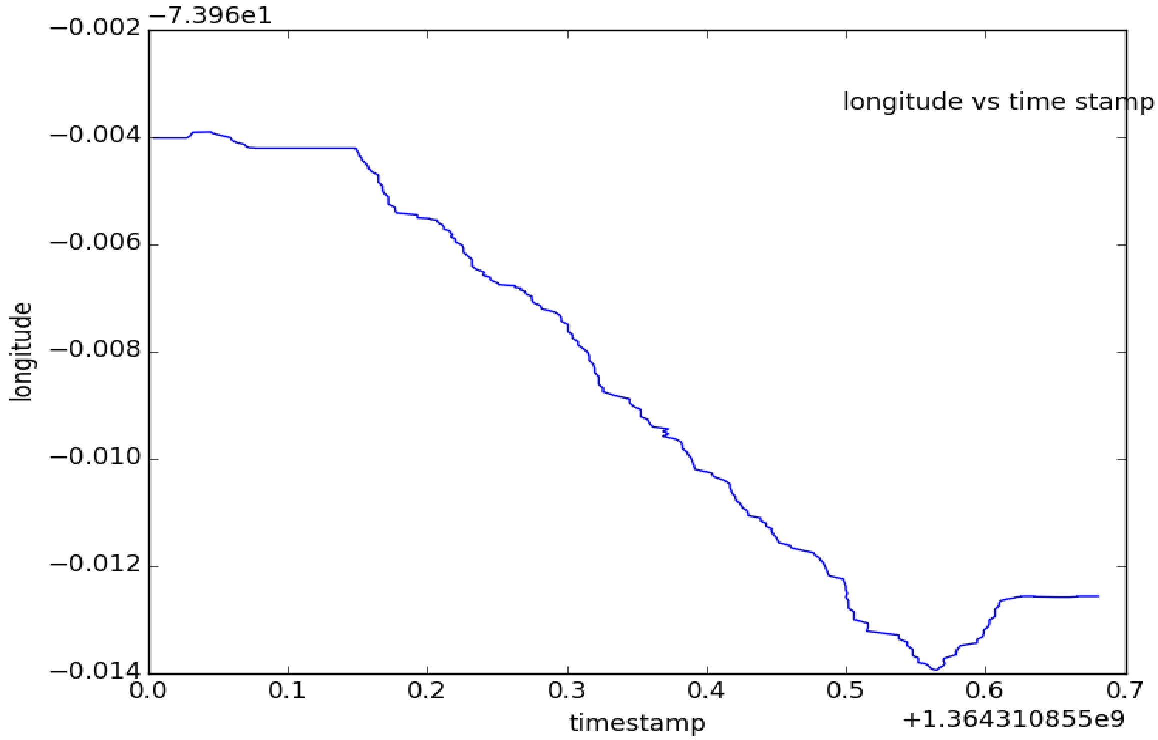
III-d) torque at transmission vs time stamp



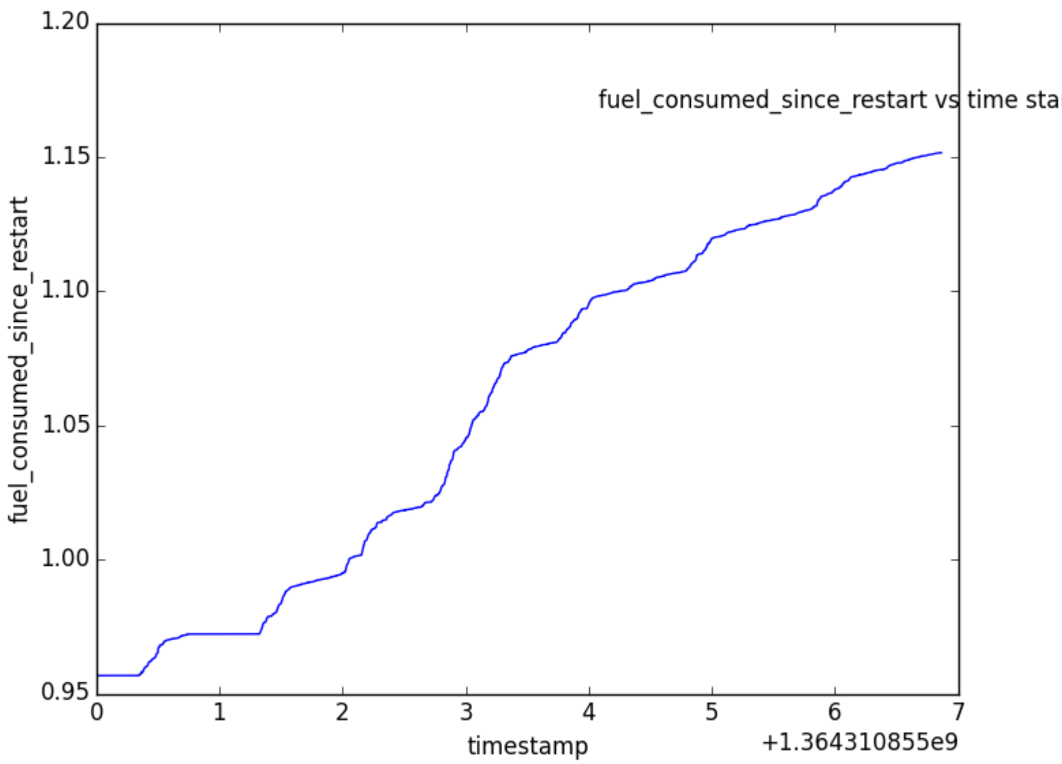
III-e) latitude vs time stamp



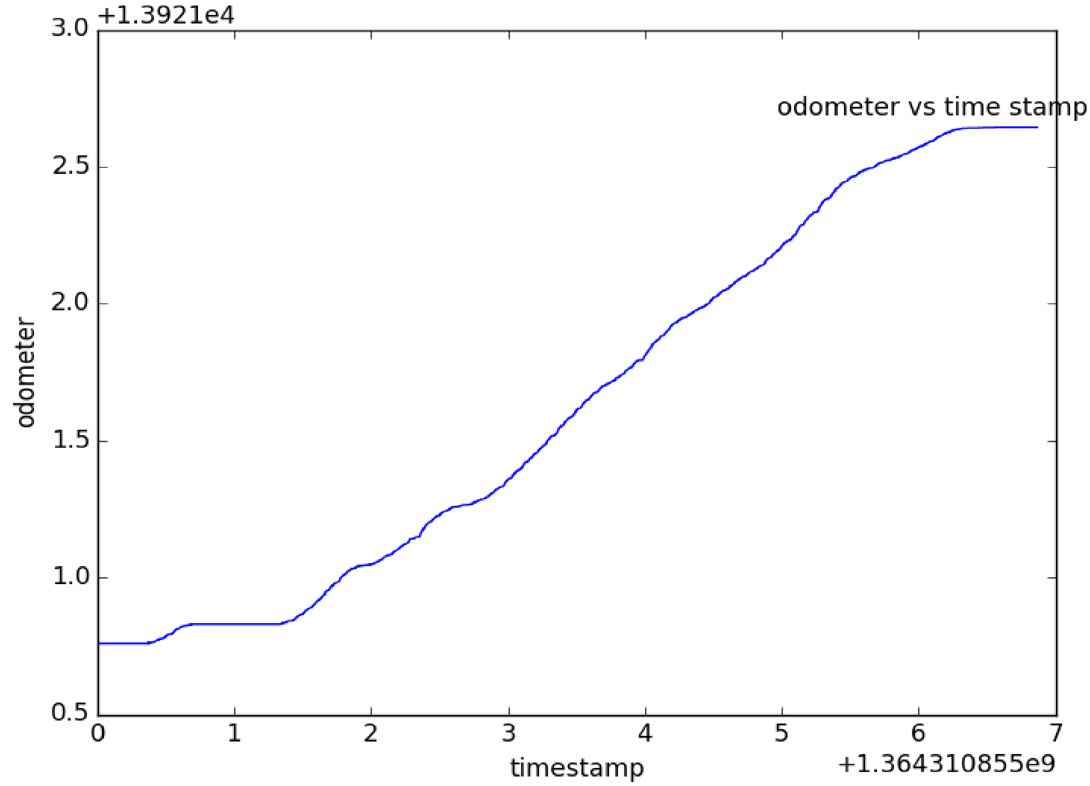
III-f) longitude vs time stamp



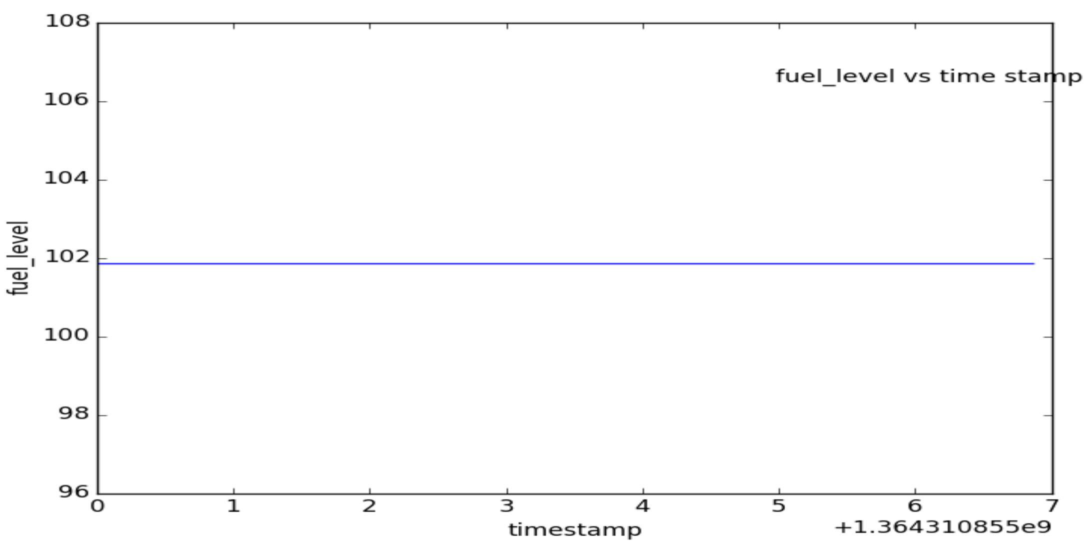
III-g) fuel consumed since restart vs time stamp



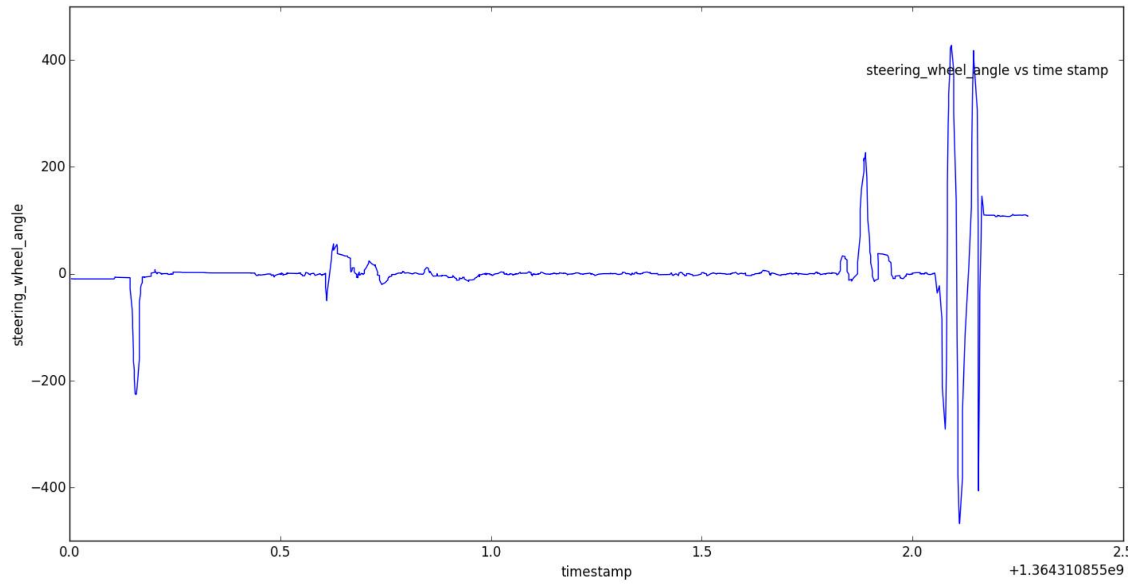
III-h) odometer reading vs time stamp



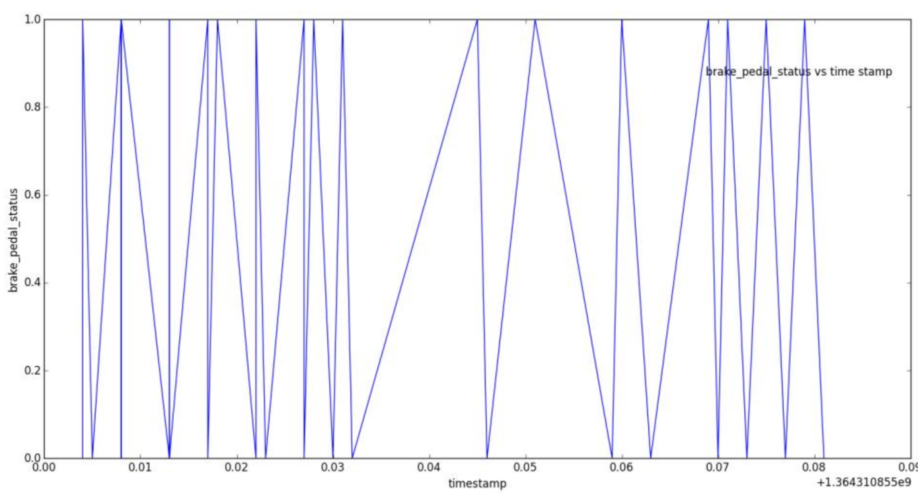
III-i) fuel level vs time stamp



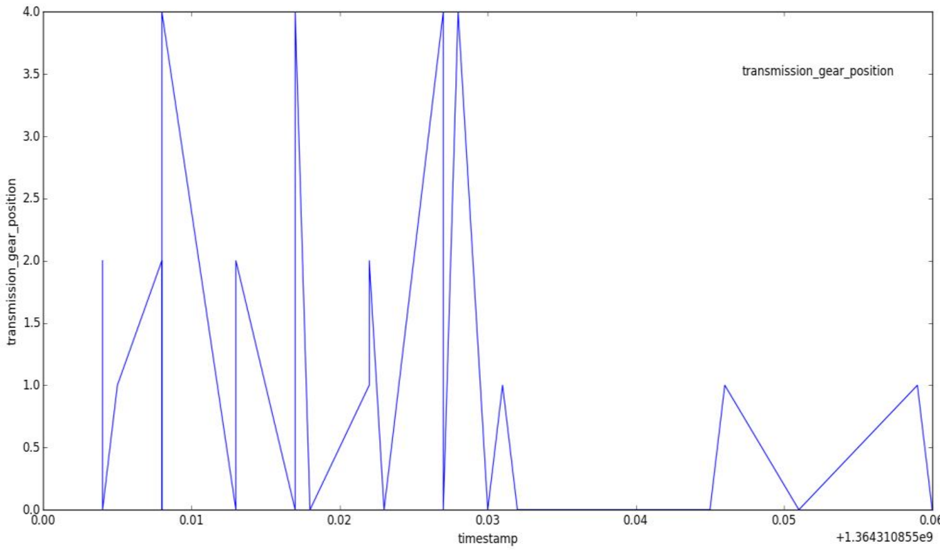
III-j) steering wheel angle vs time stamp



III-k) brake pedal status vs time stamp



III-l) transmission gear position vs time stamp



func7():

III-m) Route of Alice's vehicle according to given data

IV. <u>Observations:</u>

a) The vehicle's trip started at 104 ST W and ended at 82 ST W, according to the pygmap plot.
The vehicle stayed in idle mode, whenever its speed is zero is ~54 seconds, from the vehicle_speed vs time_stamp plot of func5().
The maximum speed of vehicle during the trip is 47.68 kilometer/hour and the average speed is 22.97 kilometer/hour. This trip took place in urban Manhattan/ New York metropolitan area.

b) The Safe driving parameters determined any motor vehicle authority/ insurance company are Speed, Cornering, Time, Acceleration & Braking. From the temporal plots obtained, it is clearly evident that Speed is moderate, braking is under control and 296(~5 minutes) to traverse 1.2 miles is very safe driving. Hence Alice's driving can be considered as safe driving.