thesis_test

September 29, 2019

1 Imports

```
In [1]: import pandas as pd
        import psycopg2

        %matplotlib inline

        import matplotlib
        import numpy as np
        import matplotlib.pyplot as plt
        from datetime import datetime
        from sklearn import neighbors, datasets
        from matplotlib.colors import ListedColormap
```

2 Conect to db

```
In [2]: # Define our connection string
        conn_string = "host='localhost' port='5432' dbname='GTA' user='jcostamagna'"

        # print the connection string we will use to connect
        print ("Connecting to database\n        ->")

        # get a connection, if a connect cannot be made an exception will be raised here
        conn = psycopg2.connect(conn_string)

        # conn.cursor will return a cursor object, you can use this cursor to perform queries
        cursor = conn.cursor()
        print( "Connected!\n" )

        #cursor.execute('SELECT * FROM  public."GTA_DataView";')
```

```
Connecting to database
        ->
Connected!
```

3 Create dataframes

```
In [3]: cursor.execute('SELECT time, idsession, scenario,' +
                       ' MAX(CASE data_type_id WHEN 1 THEN value END) as posX,' +
                       'MAX(CASE data_type_id WHEN 2 THEN value END) as posY,' +
                       ' MAX(CASE data_type_id WHEN 10 THEN value END) as hadCollision,'+
                       ' MAX(CASE data_type_id WHEN 17 THEN value END) as pedId,'+
                       ' MAX(CASE data_type_id WHEN 15 THEN value END) as nextPedRunning,'+
                       ' MAX(CASE data_type_id WHEN 11 THEN value END) as distancePed,'+
                        ' MAX(CASE data_type_id WHEN 8 THEN value END) as speed,'+
                        ' MAX(CASE data_type_id WHEN 12 THEN value END) as posPedX,'+
                        ' MAX(CASE data_type_id WHEN 13 THEN value END) as posPedY,'+
                       ' MAX(CASE data_type_id WHEN 596 THEN value END) as currentDistance'+
                       ' FROM datagta gta WHERE gta.idsession >= 144 AND gta.data_type_id in (
                       ' AND gta.idsession not in (145,146,147,148,149,151,155,158,162,164)'+
                       ' GROUP BY time, idsession, scenario;')
        rows = cursor.fetchall()
        timestamps = [row[0] for row in rows]
        if (len(timestamps) == 0):
            print ("Something went wrong")
        firstTime = timestamps[:1][0]
        lastTime = timestamps[-1:][0]
        dfGTA = pd.DataFrame(rows, index=timestamps)
        #dfDiffGTA = dfGTA.diff()
        dfGTA.columns=['time','session_id','scenario', 'Pos_X', 'Pos_Y', 'hadCollision', 'pedId

In [4]: cursor.execute('SELECT time, idsession, scenario,' +
                       ' MAX(CASE data_type_id WHEN 1 THEN value END) as Steering,' +
                       'MAX(CASE data_type_id WHEN 2 THEN value END) as Brake,' +
                       ' MAX(CASE data_type_id WHEN 3 THEN value END) as Throttle '+
                       ' FROM datasteering gta WHERE gta.idsession >= 144 AND gta.data_type_id
                       ' AND gta.idsession not in (145,146,147,148,149,151,155,158,162,164) '+
                       ' GROUP BY time, idsession, scenario;')
        rows = cursor.fetchall()
        timestamps = [row[0] for row in rows]
        df_steer = pd.DataFrame(rows, index=timestamps)
        df_steer.columns=['time2','session_id','scenario', 'steering', 'brake',  'throttle']
        freq_resample = '50000U'
        df_steer = df_steer.resample(freq_resample).ffill()
        freq_resample = '50000U'
        dfGTA = dfGTA.resample(freq_resample).ffill()
        steer = df_steer.drop(df_steer.columns[[1, 2]], axis=1)
        df_joined = pd.concat([dfGTA, steer], axis=1, join='inner')

In [5]: #pedCross = dfGTA[(dfGTA.nextPedRunning == True) & (dfGTA.distancePed < 20)  & (dfGTA.
        #df_joined['acceleration'] = df_joined['speed'].diff()
        df_joined = df_joined[~df_joined.index.duplicated()]
        df_joined.head()
```

```
Out[5]:                                               time  session_id  scenario  \
        2017-10-04 12:46:18.050                        NaT         NaN       NaN
        2017-10-04 12:46:18.100 2017-10-04 12:46:18.071232       144.0       7.0
        2017-10-04 12:46:18.150 2017-10-04 12:46:18.071232       144.0       7.0
        2017-10-04 12:46:18.200 2017-10-04 12:46:18.071232       144.0       7.0
        2017-10-04 12:46:18.250 2017-10-04 12:46:18.071232       144.0       7.0

                                    Pos_X     Pos_Y  hadCollision  pedId  \
        2017-10-04 12:46:18.050       NaN       NaN           NaN    NaN
        2017-10-04 12:46:18.100 -975.7495  82.24962           0.0   17.0
        2017-10-04 12:46:18.150 -975.7495  82.24962           0.0   17.0
        2017-10-04 12:46:18.200 -975.7495  82.24962           0.0   17.0
        2017-10-04 12:46:18.250 -975.7495  82.24962           0.0   17.0

                                 nextPedRunning  distancePed  speed   posPedX  \
        2017-10-04 12:46:18.050             NaN          NaN    NaN       NaN
        2017-10-04 12:46:18.100             0.0    70.352844    0.0  154.0415
        2017-10-04 12:46:18.150             0.0    70.352844    0.0  154.0415
        2017-10-04 12:46:18.200             0.0    70.352844    0.0  154.0415
        2017-10-04 12:46:18.250             0.0    70.352844    0.0  154.0415

                                  posPedY  currentDistance                      time2  \
        2017-10-04 12:46:18.050       NaN              NaN 2017-10-04 12:46:18.048674
        2017-10-04 12:46:18.100 -23.48739       1253.50293 2017-10-04 12:46:18.088780
        2017-10-04 12:46:18.150 -23.48739       1253.50293 2017-10-04 12:46:18.148940
        2017-10-04 12:46:18.200 -23.48739       1253.50293 2017-10-04 12:46:18.188544
        2017-10-04 12:46:18.250 -23.48739       1253.50293 2017-10-04 12:46:18.248732

                                 steering    brake  throttle
        2017-10-04 12:46:18.050   33096.0  65535.0   65535.0
        2017-10-04 12:46:18.100   33096.0  65535.0   65535.0
        2017-10-04 12:46:18.150   33096.0  65535.0   65535.0
        2017-10-04 12:46:18.200   33096.0  65535.0   65535.0
        2017-10-04 12:46:18.250   33096.0  65535.0   65535.0
```

4 Backup Csv

```
In [2]: data = pd.read_csv("DataJoined.csv", index_col=0)
        data = data.dropna()
```

```
In [3]: df_joined = data.copy()
        df_joined = df_joined[(df_joined.pedId != 16)  & (df_joined.pedId != 17)  & (df_joined
        df_joined.head()
```

```
Out[3]:                                                time  session_id  scenario  \
        2017-10-04 12:50:28.400  2017-10-04 12:50:28.383484       144.0       1.0
        2017-10-04 12:50:28.450  2017-10-04 12:50:28.437127       144.0       1.0
        2017-10-04 12:50:28.500  2017-10-04 12:50:28.494279       144.0       1.0
```

```
2017-10-04 12:50:28.550  2017-10-04 12:50:28.520850       144.0       1.0
2017-10-04 12:50:28.600  2017-10-04 12:50:28.584519       144.0       1.0

                              Pos_X      Pos_Y  hadCollision  pedId  \
2017-10-04 12:50:28.400 -975.703700  82.260376           0.0    1.0
2017-10-04 12:50:28.450 -975.709534  82.251080           0.0    1.0
2017-10-04 12:50:28.500 -975.714233  82.246090           0.0    1.0
2017-10-04 12:50:28.550 -975.716000  82.244156           0.0    1.0
2017-10-04 12:50:28.600 -975.718100  82.241540           0.0    1.0

                         nextPedRunning  distancePed     speed   posPedX  \
2017-10-04 12:50:28.400             0.0    89.979675  0.013278 -885.6892
2017-10-04 12:50:28.450             0.0    89.984790  0.135856 -885.6892
2017-10-04 12:50:28.500             0.0    89.988880  0.125762 -885.6892
2017-10-04 12:50:28.550             0.0    89.990330  0.102796 -885.6892
2017-10-04 12:50:28.600             0.0    89.991715  0.047856 -885.6892

                          posPedY  currentDistance  \
2017-10-04 12:50:28.400  70.21315       1253.45900
2017-10-04 12:50:28.450  70.21315       1253.46400
2017-10-04 12:50:28.500  70.21315       1253.46814
2017-10-04 12:50:28.550  70.21315       1253.46960
2017-10-04 12:50:28.600  70.21315       1253.47168

                                             time.1  steering    brake  \
2017-10-04 12:50:28.400  2017-10-04 12:50:28.384988   32307.0  65535.0
2017-10-04 12:50:28.450  2017-10-04 12:50:28.444647   32307.0  65535.0
2017-10-04 12:50:28.500  2017-10-04 12:50:28.484754   32307.0  65535.0
2017-10-04 12:50:28.550  2017-10-04 12:50:28.544914   32307.0  65535.0
2017-10-04 12:50:28.600  2017-10-04 12:50:28.585020   32307.0  65535.0

                         throttle
2017-10-04 12:50:28.400   65535.0
2017-10-04 12:50:28.450   65535.0
2017-10-04 12:50:28.500   65535.0
2017-10-04 12:50:28.550   65535.0
2017-10-04 12:50:28.600   65535.0
```

4.1 Clean Data

```
In [4]: df_joined['steering'] = df_joined['steering'].divide(65535)
        df_joined['brake'] = df_joined['brake'].divide(65535)
        df_joined['throttle'] = df_joined['throttle'].divide(65535)
        df_joined['acceleration'] = df_joined['speed'].diff()
        df_joined = df_joined[(df_joined.pedId != 16)  & (df_joined.pedId != 17)  & (df_joined
```

5 Means fields

```
In [5]: means = df_joined.groupby(['session_id', 'scenario', 'pedId'], as_index=False).mean()
        #clean the ones in which didnt run the pedestrian
        means = means[means.nextPedRunning != 0]
        means_cleaned = means.drop(means.columns[[3,4,6,7,9,10,11]], axis=1)

        #means_cleaned = means.dropna(thresh=11)
        #df_joined = df_joined[df_joined.notnull()]
        #session_id        scenario        pedId        hadCollision speed                stee
        ##means_cleaned['hadCollision'] = np.where(means_cleaned['hadCollision'] == 0, False,

        #means_cleaned.drop(means_cleaned[means_cleaned.nextPedRunning == 0])
        means_cleaned.head()
```

```
Out[5]:    session_id  scenario  pedId  hadCollision      speed  steering     brake  \
        0       144.0       1.0    1.0      0.000000   5.919151  0.503649  0.965743
        1       144.0       1.0    3.0      0.000000   7.580378  0.499771  0.891302
        2       144.0       1.0   15.0      0.008366   9.474048  0.494557  0.952182
        3       144.0       2.0    0.0      0.000000   3.398595  0.523305  0.960227
        4       144.0       2.0    2.0      0.000000  11.669419  0.500661  0.891913

           throttle  acceleration
        0  0.820576      0.030731
        1  0.878839     -0.026652
        2  0.781126      0.006292
        3  0.858795      0.027363
        4  0.522365      0.008028
```

```
In [6]: means_cleaned.shape
```

```
Out[6]: (655, 9)
```

```
In [7]: #Rows in which the pedestrian didnt run
        means[means.nextPedRunning == 0]
```

```
Out[7]: Empty DataFrame
        Columns: [session_id, scenario, pedId, Pos_X, Pos_Y, hadCollision, nextPedRunning, dis
        Index: []
```

6 Variances

```
In [8]: variance = df_joined.groupby(['session_id', 'scenario', 'pedId'], as_index=False).var(
        variance_cleaned = variance.drop(variance.columns[[3,4,5,6,7,9,10,11]], axis=1)
        #variance_cleaned['hadCollision'] = np.where(variance_cleaned['hadCollision'] == 0, Fa
        variance_cleaned.head()
```

```
Out[8]:    session_id  scenario  pedId      speed  steering     brake  throttle  \
        0       144.0       1.0    1.0  13.796202  0.000655  0.014468  0.028719
```

```
1       144.0       1.0    3.0  31.451253  0.000345  0.058767  0.010391
2       144.0       1.0   15.0  53.873833  0.001231  0.022506  0.045416
3       144.0       2.0    0.0   5.036717  0.000101  0.018643  0.075860
4       144.0       2.0    2.0  47.209285  0.000396  0.055982  0.112551

   acceleration
0      0.039370
1      0.063480
2      0.106281
3      0.027735
4      0.159198
```

```
In [9]: variance_cleaned.shape

Out[9]: (671, 8)
```

7 Between start, run and crash

7.1 Total time

```
In [10]: first = df_joined.groupby(['session_id', 'scenario', 'pedId'], as_index=False).first(
         last = df_joined.groupby(['session_id', 'scenario', 'pedId'], as_index=False).last()

In [11]: total_time = first[['session_id', 'scenario', 'pedId']].copy()
         first['timeFinal'] = last['time']
         total_time['total_time'] = (pd.to_datetime(first['timeFinal']) - pd.to_datetime(first
         #plt.plot(first['total_time'])
         #s.dt.total_seconds()
         total_time.head()

Out[11]:    session_id  scenario  pedId  total_time
         0        144.0       1.0    1.0   15.405173
         1        144.0       1.0    3.0   11.412381
         2        144.0       1.0   15.0  102.356492
         3        144.0       2.0    0.0   14.315100
         4        144.0       2.0    2.0    7.505478
```

7.2 Initial distance

```
In [12]: initial_distance = first[['session_id', 'scenario', 'pedId', 'distancePed']].copy()
         initial_distance.head()
         ## desconfianza con el index 2 y su tiempo
         #144.0        1.0        15.0

         #testing_distance = df_joined[(df_joined.pedId == 15)  & (df_joined.scenario == 1)  &
         #plt.plot(testing_distance['distancePed'])

Out[12]:    session_id  scenario  pedId  distancePed
         0        144.0       1.0    1.0    89.979675
```

```
1       144.0       1.0    3.0    11.443494
2       144.0       1.0   15.0    10.211066
3       144.0       2.0    0.0    47.977978
4       144.0       2.0    2.0     3.084074
```

```
In [13]: max_values = df_joined.groupby(['session_id', 'scenario', 'pedId'], as_index=False).ma
         initial_distance = max_values[['session_id', 'scenario', 'pedId', 'distancePed']].copy
         initial_distance.head()
```

```
Out[13]:    session_id  scenario  pedId  distancePed
         0       144.0       1.0    1.0    89.992450
         1       144.0       1.0    3.0    85.063860
         2       144.0       1.0   15.0   789.212800
         3       144.0       2.0    0.0    47.977978
         4       144.0       2.0    2.0    88.011610
```

7.3 Max Speed

```
In [14]: max_speed = max_values[['session_id', 'scenario', 'pedId', 'speed']].copy()
         max_speed.head()
```

```
Out[14]:    session_id  scenario  pedId      speed
         0       144.0       1.0    1.0  11.669766
         1       144.0       1.0    3.0  13.499710
         2       144.0       1.0   15.0  25.851397
         3       144.0       2.0    0.0  10.266865
         4       144.0       2.0    2.0  20.055070
```

```
In [15]: max_speed.shape
```

```
Out[15]: (671, 4)
```

7.4 Before pedestrian crossed

```
In [16]: def delete_unused_columns(df):
             df.drop('Pos_X', axis=1, inplace=True, errors='ignore')
             df.drop('Pos_Y', axis=1, inplace=True, errors='ignore')
             df.drop('distancePed', axis=1, inplace=True, errors='ignore')
             df.drop('nextPedRunning', axis=1, inplace=True, errors='ignore')
             df.drop('posPedX', axis=1, inplace=True, errors='ignore')
             df.drop('posPedY', axis=1, inplace=True, errors='ignore')
             df.drop('currentDistance', axis=1, inplace=True, errors='ignore')
             df.drop('hadCollision', axis=1, inplace=True, errors='ignore')
```

```
In [17]: pedDidNotCross = df_joined[(df_joined.nextPedRunning == False)]
         firstRowPedDidNotCross = pedDidNotCross.groupby(['session_id', 'scenario', 'pedId'], a
         lastRowPedDidNotCross = pedDidNotCross.groupby(['session_id', 'scenario', 'pedId'], as
         meansPedDidNotCross = pedDidNotCross.groupby(['session_id', 'scenario', 'pedId'], as_i
         varPedDidNotCross = pedDidNotCross.groupby(['session_id', 'scenario', 'pedId'], as_ind
         delete_unused_columns(meansPedDidNotCross)
         delete_unused_columns(varPedDidNotCross)
```

```
In [18]: meansPedDidNotCross.shape

Out[18]: (652, 8)

In [19]: meansPedDidNotCross[meansPedDidNotCross.isnull().values == True]

Out[19]: Empty DataFrame
         Columns: [session_id, scenario, pedId, speed, steering, brake, throttle, acceleration]
         Index: []

In [20]: varPedDidNotCross = varPedDidNotCross.dropna()
         varPedDidNotCross.shape

Out[20]: (648, 8)

In [21]: varPedDidNotCross[varPedDidNotCross.isnull().values == True]

Out[21]: Empty DataFrame
         Columns: [session_id, scenario, pedId, speed, steering, brake, throttle, acceleration]
         Index: []
```

7.4.1 Total time before run

```
In [22]: beforePedCross = firstRowPedDidNotCross[['session_id', 'scenario', 'pedId']].copy()
         firstRowPedDidNotCross['timeFinal'] = lastRowPedDidNotCross['time']
         beforePedCross['total_time_before_run'] = (pd.to_datetime(firstRowPedDidNotCross['time
         beforePedCross.head()
         beforePedCross.shape

Out[22]: (652, 4)

In [23]: beforePedCrossMerge = pd.merge(varPedDidNotCross, meansPedDidNotCross , on=['session_
         delete_unused_columns(beforePedCrossMerge)
         beforePedCrossMerge.shape

Out[23]: (648, 13)

In [24]: beforePedCrossTotalColumns = pd.merge(beforePedCrossMerge, beforePedCross , on=['sess
         beforePedCrossTotalColumns.head()

Out[24]:    session_id  scenario  pedId  speed_before_var  steering_before_var  \
         0       144.0       1.0    1.0         13.894033             0.000338
         1       144.0       1.0    3.0          1.185339             0.000238
         2       144.0       1.0   15.0         54.290571             0.001118
         3       144.0       2.0    0.0          4.118861             0.000045
         4       144.0       2.0    2.0          6.272065             0.000233

            brake_before_var  throttle_before_var  acceleration_before_var  \
         0          0.000000             0.017802                 0.016136
         1          0.000000             0.013602                 0.004440
```

```
2          0.009412             0.045813                 0.032457
3          0.020263             0.014640                 0.019610
4          0.000000             0.094616                 0.011947

   speed_before_mean  steering_before_mean  brake_before_mean  \
0           6.614995              0.517642           1.000000
1          12.405217              0.481101           1.000000
2           9.719554              0.496611           0.969863
3           3.134071              0.525620           0.956432
4          15.771787              0.492397           1.000000

   throttle_before_mean  acceleration_before_mean  total_time_before_run
0              0.863178                  0.038096              10.091857
1              0.772151                  0.054448               3.673778
2              0.788247                  0.013562              94.469000
3              0.933556                  0.000010              13.052240
4              0.310690                  0.178377               2.488626
```

```
In [25]: beforePedCrossTotalColumns.shape
```

```
Out[25]: (648, 14)
```

7.5 When pedestrian is running

```
In [26]: pedCross = df_joined[(df_joined.nextPedRunning == True)]
```

```
In [27]: firstRowPedCross = pedCross.groupby(['session_id', 'scenario', 'pedId'], as_index=Fals
         lastRowPedCross = pedCross.groupby(['session_id', 'scenario', 'pedId'], as_index=False
         meansPedCross = pedCross.groupby(['session_id', 'scenario', 'pedId'], as_index=False)
         varPedCross = pedCross.groupby(['session_id', 'scenario', 'pedId'], as_index=False).va
         delete_unused_columns(meansPedCross)
         delete_unused_columns(varPedCross)
```

7.5.1 Total time when running

```
In [28]: whenPedCross = firstRowPedCross[['session_id', 'scenario', 'pedId']].copy()
         firstRowPedCross['timeFinal'] = lastRowPedCross['time']
         whenPedCross['total_time_when_running'] = (pd.to_datetime(firstRowPedCross['timeFinal
         whenPedCross.head()
```

```
Out[28]:    session_id  scenario  pedId  total_time_when_running
         0       144.0       1.0    1.0                 5.239119
         1       144.0       1.0    3.0                 7.705014
         2       144.0       1.0   15.0                 7.848389
         3       144.0       2.0    0.0                 1.202199
         4       144.0       2.0    2.0                 4.975241
```

```
In [29]: whenPedCrossMerge = pd.merge(varPedCross, meansPedCross , on=['session_id','scenario'
         delete_unused_columns(whenPedCrossMerge)
         whenPedCrossTotalColumns = pd.merge(whenPedCrossMerge, whenPedCross , on=['session_id
         whenPedCrossTotalColumns.head()
```

```
Out[29]:    session_id  scenario  pedId  speed_when_running_var  \
         0       144.0       1.0    1.0               11.033679
         1       144.0       1.0    3.0               29.475829
         2       144.0       1.0   15.0               42.756716
         3       144.0       2.0    0.0                6.459006
         4       144.0       2.0    2.0               55.270252

            steering_when_running_var  brake_when_running_var  \
         0                   0.000174                0.035741
         1                   0.000151                0.078618
         2                   0.001915                0.121984
         3                   0.000048                0.000000
         4                   0.000429                0.075733

            throttle_when_running_var  acceleration_when_running_var  \
         0                   0.039745                       0.083699
         1                   0.000856                       0.087210
         2                   0.035585                       0.869046
         3                   0.044336                       0.023489
         4                   0.087892                       0.212602

            speed_when_running_mean  steering_when_running_mean  \
         0                 4.593110                    0.476982
         1                 5.276907                    0.508684
         2                 6.933534                    0.473303
         3                 6.170808                    0.499042
         4                 9.577210                    0.504876

            brake_when_running_mean  throttle_when_running_mean  \
         0                 0.900460                    0.739390
         1                 0.839408                    0.929774
         2                 0.769213                    0.707442
         3                 1.000000                    0.075305
         4                 0.836788                    0.630319

            acceleration_when_running_mean  total_time_when_running
         0                        0.016767                 5.239119
         1                       -0.065371                 7.705014
         2                       -0.068940                 7.848389
         3                        0.314027                 1.202199
         4                       -0.078850                 4.975241
```

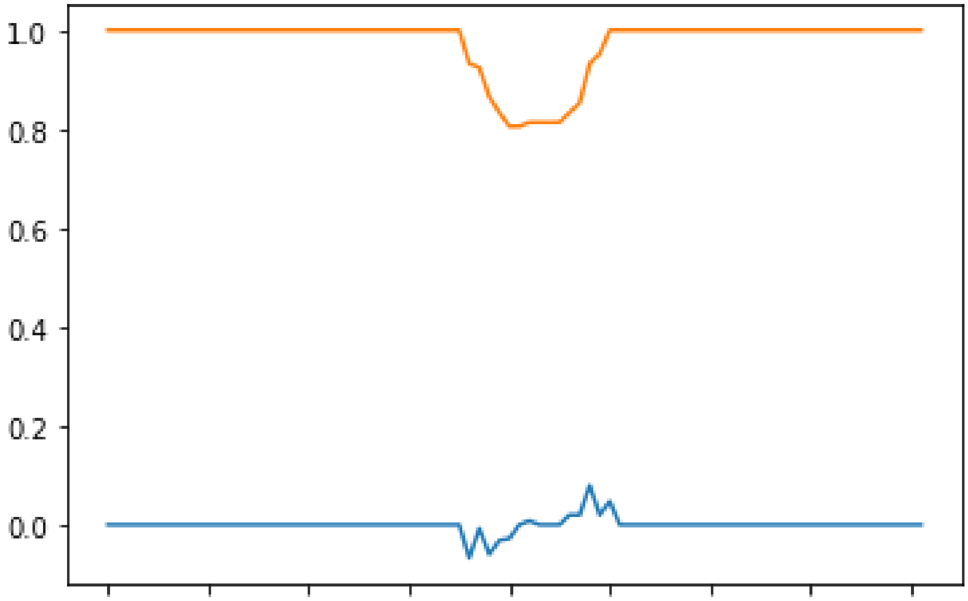
```
In [30]: whenPedCrossTotalColumns.shape
```

```
Out[30]: (655, 14)
```

```
In [31]: whenPedCrossTotalColumns.isnull().values.any()

Out[31]: False
```

7.5.2 Break Reaction Time

```
In [81]: df_joined['brakeDiff'] = df_joined['brake'].diff()
         pedCross = df_joined[(df_joined.nextPedRunning == True)]
         dfNew = pedCross[(pedCross.session_id == 175.0) & (pedCross.scenario == 6.0) & (pedCro
         dfNew['brakeDiff'].plot()
         dfNew['brake'].plot()

Out[81]: <matplotlib.axes._subplots.AxesSubplot at 0x1112cabe0>
```



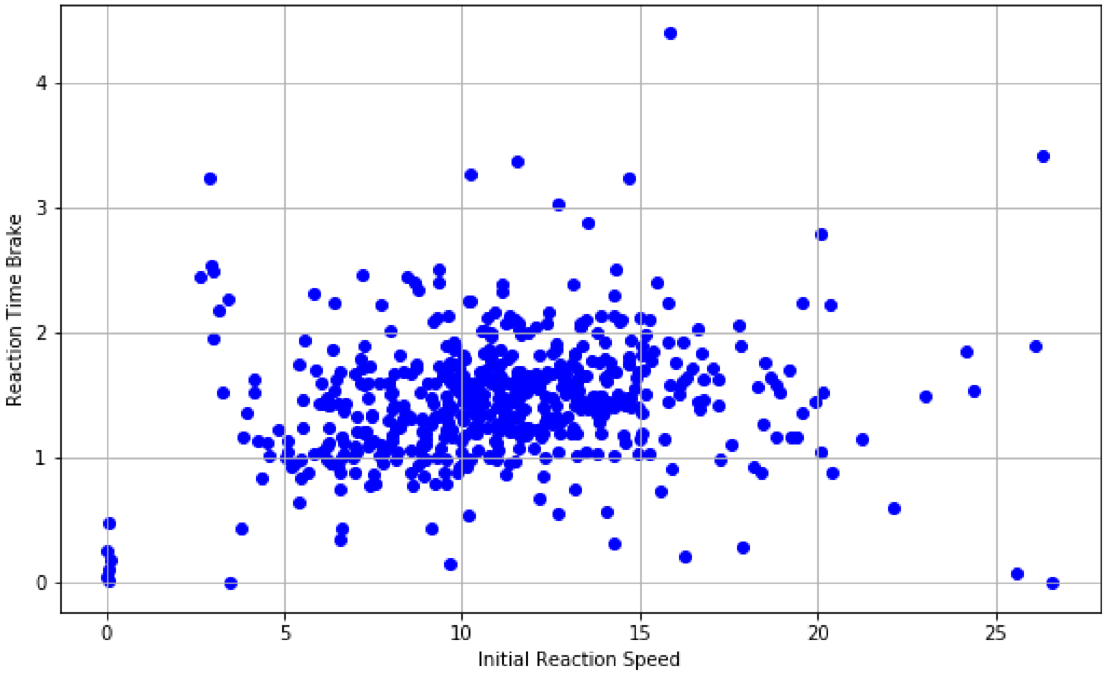
```
In [82]: startReactionBrake = pedCross
         finishReactionBrake = pedCross[(pedCross.brakeDiff < 0)]
         finishReactionBrake.shape

Out[82]: (6536, 19)

In [83]: firstRowPedCross = startReactionBrake.groupby(['session_id', 'scenario', 'pedId'], as_
         firstRowPressBrake = finishReactionBrake.groupby(['session_id', 'scenario', 'pedId'],
         firstRowPressBrake.shape
         firstRowPedCross.shape

Out[83]: (655, 19)

In [84]: reactionBrake = pd.merge(firstRowPedCross, firstRowPressBrake, on=['session_id','scena
         reactionBrake.head()
```

```
Out[84]:    session_id  scenario  pedId                   time_init  Pos_X_init  \
         0       144.0       1.0    1.0  2017-10-04 12:50:38.549538 -909.306458
         1       144.0       1.0    3.0  2017-10-04 12:50:47.530114 -839.965700
         2       144.0       1.0   15.0  2017-10-04 12:52:29.782335  -62.199753
         3       144.0       2.0    0.0  2017-10-04 12:57:42.500777 -933.749300
         4       144.0       2.0    2.0  2017-10-04 12:57:46.323453 -885.836365

            Pos_Y_init  hadCollision_init  nextPedRunning_init  distancePed_init  \
         0   80.473330                0.0                  1.0         23.562530
         1   83.164460                0.0                  1.0         39.895280
         2   40.680523                0.0                  1.0         49.374313
         3   84.487495                0.0                  1.0          7.474411
         4   77.199104                0.0                  1.0         49.291360

            speed_init      ...        speed_fin  posPedX_fin  posPedY_fin  \
         0    7.754880      ...         8.946986  -885.665649    72.027790
         1   13.472353      ...        13.034663  -801.989441    93.251990
         2   25.585112      ...        25.720684   -20.775450    15.126913
         3    2.517837      ...              NaN          NaN          NaN
         4   19.412086      ...        19.944923  -836.171631    80.316110

           currentDistance_fin                  time.1_fin  steering_fin  brake_fin  \
         0          1178.73486  2017-10-04 12:50:39.586297      0.477104   0.980163
         1          1094.34827  2017-10-04 12:50:49.648762      0.533089   0.980163
         2           355.46260  2017-10-04 12:52:29.893130      0.504952   0.900801
         3                 NaN                         NaN           NaN        NaN
         4          1141.20886  2017-10-04 12:57:47.485545      0.480339   0.964294

            throttle_fin  acceleration_fin brakeDiff_fin
         0      0.948409          0.072344     -0.019837
         1      0.948409         -0.009284     -0.019837
         2      0.445304          0.069757     -0.099199
         3           NaN               NaN           NaN
         4      0.932540         -0.081133     -0.035706

         [5 rows x 35 columns]
```

```
In [85]: reactionPressBrake = firstRowPedCross[['session_id', 'scenario', 'pedId', 'speed']].co
         reactionPressBrake['reaction_time'] = (pd.to_datetime(reactionBrake['time_fin']) - pd
         reactionPressBrake = reactionPressBrake.dropna()
         reactionPressBrake.head()
```

```
Out[85]:    session_id  scenario  pedId      speed  reaction_time
         0       144.0       1.0    1.0   7.754880       1.048791
         1       144.0       1.0    3.0  13.472353       2.106615
         2       144.0       1.0   15.0  25.585112       0.079211
         4       144.0       2.0    2.0  19.412086       1.161592
         5       144.0       2.0    4.0  18.461056       1.275896
```

```
In [86]: reactionPressBrake.shape

Out[86]: (573, 5)

In [87]: reactionPressBrake.isnull().values.any()

Out[87]: False

In [88]: plt.figure(figsize=(10, 6))
         plt.xlabel("Initial Reaction Speed")
         plt.ylabel("Reaction Time Brake")
         reactionPressBrake = reactionPressBrake[reactionPressBrake.reaction_time < 50]

         noHits = plt.scatter(reactionPressBrake['speed'], reactionPressBrake['reaction_time']

         plt.grid(True)

         plt.show()
```

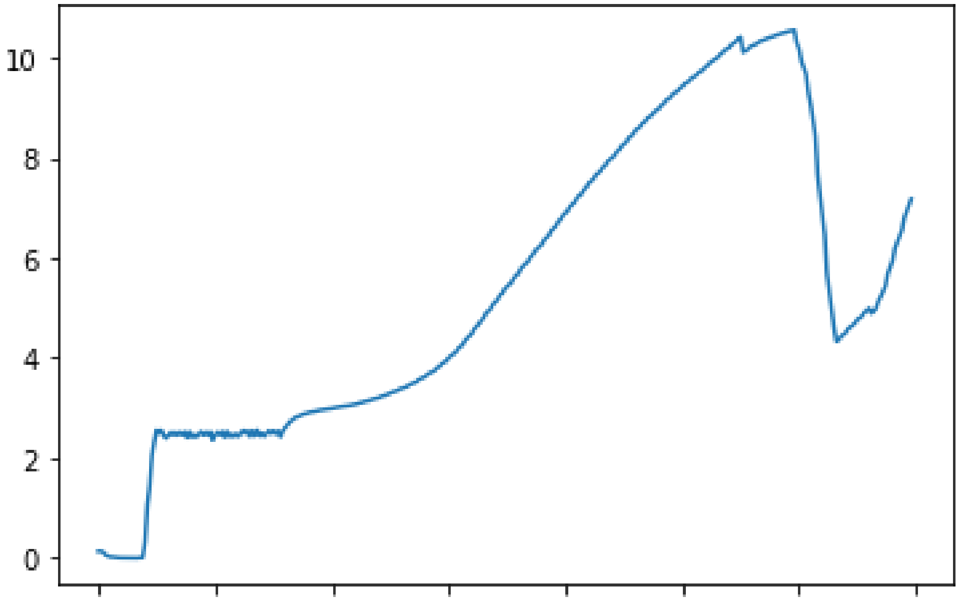


```
In [89]: reactionPressBrake.shape

Out[89]: (559, 5)
```

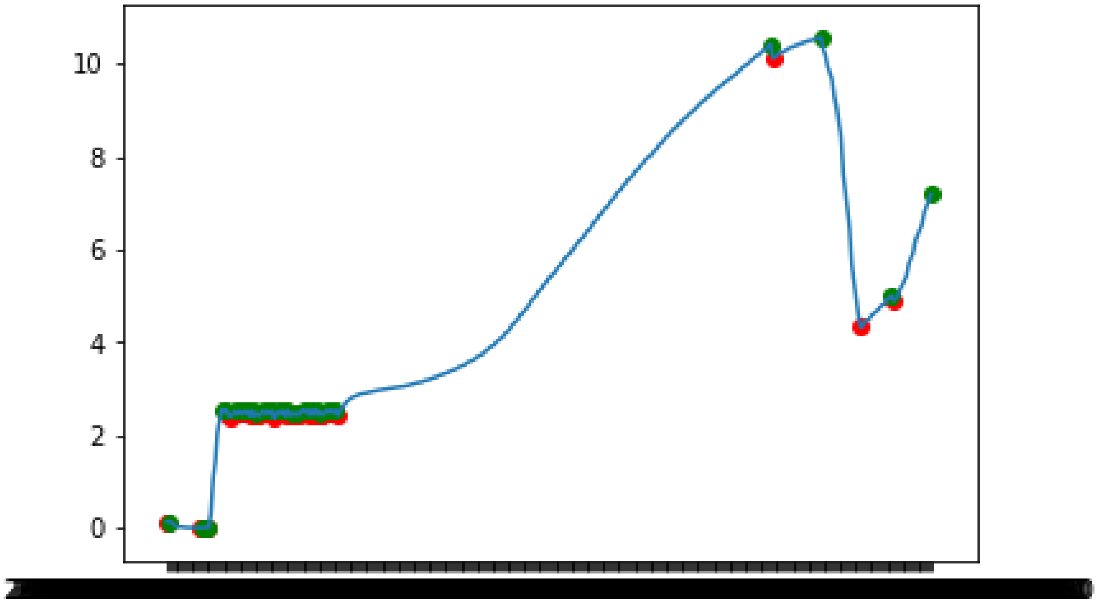
7.6 Agressive driving measure: PKE

```
In [38]: dfPKE = df_joined.copy()
         dfNew = dfPKE[(dfPKE.session_id == 175.0) & (dfPKE.scenario == 6.0) & (dfPKE.pedId ==
         dfNew['speed'].plot()
```

```
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x10cdaf2e8>
```



```
In [39]: from scipy.signal import argrelextrema

         df = dfNew.copy()
         n=1 # number of points to be checked before and after
         # Find local peaks
         df['min'] = df.iloc[argrelextrema(df.speed.values, np.less_equal, order=n)[0]]['speed
         df['max'] = df.iloc[argrelextrema(df.speed.values, np.greater_equal, order=n)[0]]['sp

         # Plot results
         plt.scatter(df.index, df['min'], c='r')
         plt.scatter(df.index, df['max'], c='g')
         plt.plot(df.index, df['speed'])
         plt.show()
```



```
In [40]: df.head()

Out[40]:                                                 time  session_id  scenario  \
         2017-10-10 12:26:12.200  2017-10-10 12:26:12.192493       175.0       6.0
         2017-10-10 12:26:12.250  2017-10-10 12:26:12.234605       175.0       6.0
         2017-10-10 12:26:12.300  2017-10-10 12:26:12.292259       175.0       6.0
         2017-10-10 12:26:12.350  2017-10-10 12:26:12.346905       175.0       6.0
         2017-10-10 12:26:12.400  2017-10-10 12:26:12.374478       175.0       6.0

                                      Pos_X      Pos_Y  hadCollision  pedId  \
         2017-10-10 12:26:12.200 -975.707031  82.254616           0.0    1.0
         2017-10-10 12:26:12.250 -975.710000  82.250580           0.0    1.0
         2017-10-10 12:26:12.300 -975.714844  82.245700           0.0    1.0
         2017-10-10 12:26:12.350 -975.717900  82.242210           0.0    1.0
         2017-10-10 12:26:12.400 -975.718500  82.241270           0.0    1.0

                                  nextPedRunning  distancePed     speed    ...      \
         2017-10-10 12:26:12.200             0.0     89.97974  0.137321    ...
         2017-10-10 12:26:12.250             0.0     89.98381  0.140411    ...
         2017-10-10 12:26:12.300             0.0     89.98826  0.123498    ...
         2017-10-10 12:26:12.350             0.0     89.99129  0.068495    ...
         2017-10-10 12:26:12.400             0.0     89.99179  0.044158    ...

                                   posPedY  currentDistance  \
         2017-10-10 12:26:12.200  70.21315       1253.46167
         2017-10-10 12:26:12.250  70.21315       1253.46448
```

```
2017-10-10 12:26:12.300  70.21315       1253.46863
2017-10-10 12:26:12.350  70.21315       1253.47144
2017-10-10 12:26:12.400  70.21315       1253.47192

                                             time.1  steering  brake  \
2017-10-10 12:26:12.200  2017-10-10 12:26:12.195000  0.499016    1.0
2017-10-10 12:26:12.250  2017-10-10 12:26:12.235107  0.499016    1.0
2017-10-10 12:26:12.300  2017-10-10 12:26:12.295267  0.499016    1.0
2017-10-10 12:26:12.350  2017-10-10 12:26:12.334873  0.499016    1.0
2017-10-10 12:26:12.400  2017-10-10 12:26:12.395032  0.499016    1.0

                         throttle  acceleration  brakeDiff       min       max
2017-10-10 12:26:12.200       1.0     -5.898169        0.0  0.137321       NaN
2017-10-10 12:26:12.250       1.0      0.003090        0.0       NaN  0.140411
2017-10-10 12:26:12.300       1.0     -0.016913        0.0       NaN       NaN
2017-10-10 12:26:12.350       1.0     -0.055003        0.0       NaN       NaN
2017-10-10 12:26:12.400       1.0     -0.024337        0.0       NaN       NaN

[5 rows x 21 columns]
```

```
In [41]: import math

         def aggressiveDriverPKE(df):
             vi = None
             acumulative = 0
             distance = df.currentDistance.max() - df.currentDistance.min()
             df = df[(df['min'].notnull() | df['max'].notnull())]
             for index, row in df.iterrows():
                 if (not math.isnan(row['min'])):
                     vi = row['min']
                 else:
                     if (vi is None):
                         continue
                     else:
                         if (not math.isnan(row['max'])):
                             vf = row['max']
                             acumulative += vf*vf - vi * vi
                             vi = None
                             vf = None

             return acumulative/distance
```

```
In [42]: n=1 # number of points to be checked before and after
         # Find local peaks
         dfPKE = df_joined.copy()

         dfPKE['min'] = dfPKE.iloc[argrelextrema(dfPKE.speed.values, np.less_equal, order=n)[0
         dfPKE['max'] = dfPKE.iloc[argrelextrema(dfPKE.speed.values, np.greater_equal, order=n
        #print(aggressiveDriverPKE( dfPKE[(dfPKE.session_id == 175.0) & (dfPKE.scenario == 6.
```

```
In [43]: #dfPKE = dfPKE[['session_id', 'scenario', 'pedId', 'currentDistance', 'min','max']]
         #dfPKE = dfPKE[(dfPKE['min'].notnull() | dfPKE['max'].notnull())]
         #dfPKE.head()
```

```
In [44]: dfPKE = dfPKE[['session_id', 'scenario', 'pedId', 'currentDistance', 'min','max']]
         total_PKE = dfPKE.groupby(['session_id', 'scenario', 'pedId'], as_index=False).apply(a

         total_PKE.head()
```

```
Out[44]:    session_id  scenario  pedId       PKE
         0       144.0       1.0    1.0  1.932290
         1       144.0       1.0    3.0  0.878493
         2       144.0       1.0   15.0  2.857169
         3       144.0       2.0    0.0  3.659420
         4       144.0       2.0    2.0  2.969647
```

```
In [45]: total_PKE.shape
```

```
Out[45]: (671, 4)
```

7.6.1 Steering PKE

```
In [46]: dfSteeringPKE = df_joined.copy()
         dfSteeringPKE['min'] = dfSteeringPKE.iloc[argrelextrema(dfSteeringPKE.speed.values, np
         dfSteeringPKE['max'] = dfSteeringPKE.iloc[argrelextrema(dfSteeringPKE.speed.values, np
```

```
In [47]: dfSteeringPKE = dfSteeringPKE[['session_id', 'scenario', 'pedId', 'currentDistance',
         total_steering_PKE = dfSteeringPKE.groupby(['session_id', 'scenario', 'pedId'], as_inc

         total_steering_PKE.head()
```

```
Out[47]:    session_id  scenario  pedId  PKE_Steering
         0       144.0       1.0    1.0     -0.000150
         1       144.0       1.0    3.0      0.000274
         2       144.0       1.0   15.0      0.000108
         3       144.0       2.0    0.0     -0.000627
         4       144.0       2.0    2.0     -0.000258
```

```
In [48]: total_steering_PKE.shape
```

```
Out[48]: (671, 4)
```

7.7 Backup PKE's

```
In [50]: #total_PKE.to_csv('pke.csv', index=False)
         #total_steering_PKE.to_csv('steering_pke.csv', index=False)

         total_PKE = pd.read_csv("pke.csv")
         total_steering_PKE = pd.read_csv("steering_pke.csv")
```

8 Merge dataframes

```
In [90]: df_suffix = pd.merge(means_cleaned, variance_cleaned, on=['session_id','scenario','pe
         df_suffix = pd.merge(df_suffix, total_time, on=['session_id','scenario','pedId'],how=
         df_suffix = pd.merge(df_suffix, initial_distance, on=['session_id','scenario','pedId']
         df_suffix = pd.merge(df_suffix, max_speed, on=['session_id','scenario','pedId'],how='
         df_suffix = pd.merge(df_suffix, total_PKE, on=['session_id','scenario','pedId'],how='
         df_suffix = pd.merge(df_suffix, total_steering_PKE, on=['session_id','scenario','pedI
         df_suffix = pd.merge(df_suffix, reactionPressBrake, on=['session_id','scenario','pedI
         df_suffix.head()
```

```
Out[90]:    session_id  scenario  pedId  hadCollision  speed_total_mean  \
         0       144.0       1.0    1.0      0.000000          5.919151
         1       144.0       1.0    3.0      0.000000          7.580378
         2       144.0       1.0   15.0      0.008366          9.474048
         3       144.0       2.0    2.0      0.000000         11.669419
         4       144.0       2.0    4.0      0.000000         12.187044

            steering_total_mean  brake_total_mean  throttle_total_mean  \
         0             0.503649          0.965743             0.820576
         1             0.499771          0.891302             0.878839
         2             0.494557          0.952182             0.781126
         3             0.500661          0.891913             0.522365
         4             0.499769          0.861132             0.558120

            acceleration_total_mean  speed_total_var      ...        brake_total_var  \
         0                 0.030731        13.796202      ...               0.014468
         1                -0.026652        31.451253      ...               0.058767
         2                 0.006292        53.873833      ...               0.022506
         3                 0.008028        47.209285      ...               0.055982
         4                 0.001881        42.031423      ...               0.102442

            throttle_total_var  acceleration_total_var  total_time  distancePed  \
         0            0.028719                0.039370   15.405173    89.992450
         1            0.010391                0.063480   11.412381    85.063860
         2            0.045416                0.106281  102.356492   789.212800
         3            0.112551                0.159198    7.505478    88.011610
         4            0.079023                0.158822    8.681609   105.973686

                speed       PKE  PKE_Steering  speed_react  reaction_time
         0  11.669766  1.932290     -0.000150     7.754880       1.048791
         1  13.499710  0.878493      0.000274    13.472353       2.106615
         2  25.851397  2.857169      0.000108    25.585112       0.079211
         3  20.055070  2.969647     -0.000258    19.412086       1.161592
         4  19.697004  4.033468      0.000066    18.461056       1.275896

         [5 rows x 21 columns]
```

```
In [91]: def delete_common_columns(df):
```

```
                df.drop('session_id', axis=1, inplace=True, errors='ignore')
                df.drop('pedId', axis=1, inplace=True, errors='ignore')

        delete_common_columns(df_suffix)
        df_scenario_cluster = df_suffix.copy()
        df_suffix.drop('scenario', axis=1, inplace=True, errors='ignore')
        df_suffix.rename(columns={'speed':'max_speed'}, inplace=True)
```

```
In [92]: df_suffix.isnull().values.any()
```

```
Out[92]: False
```

```
In [93]: df_suffix.shape
```

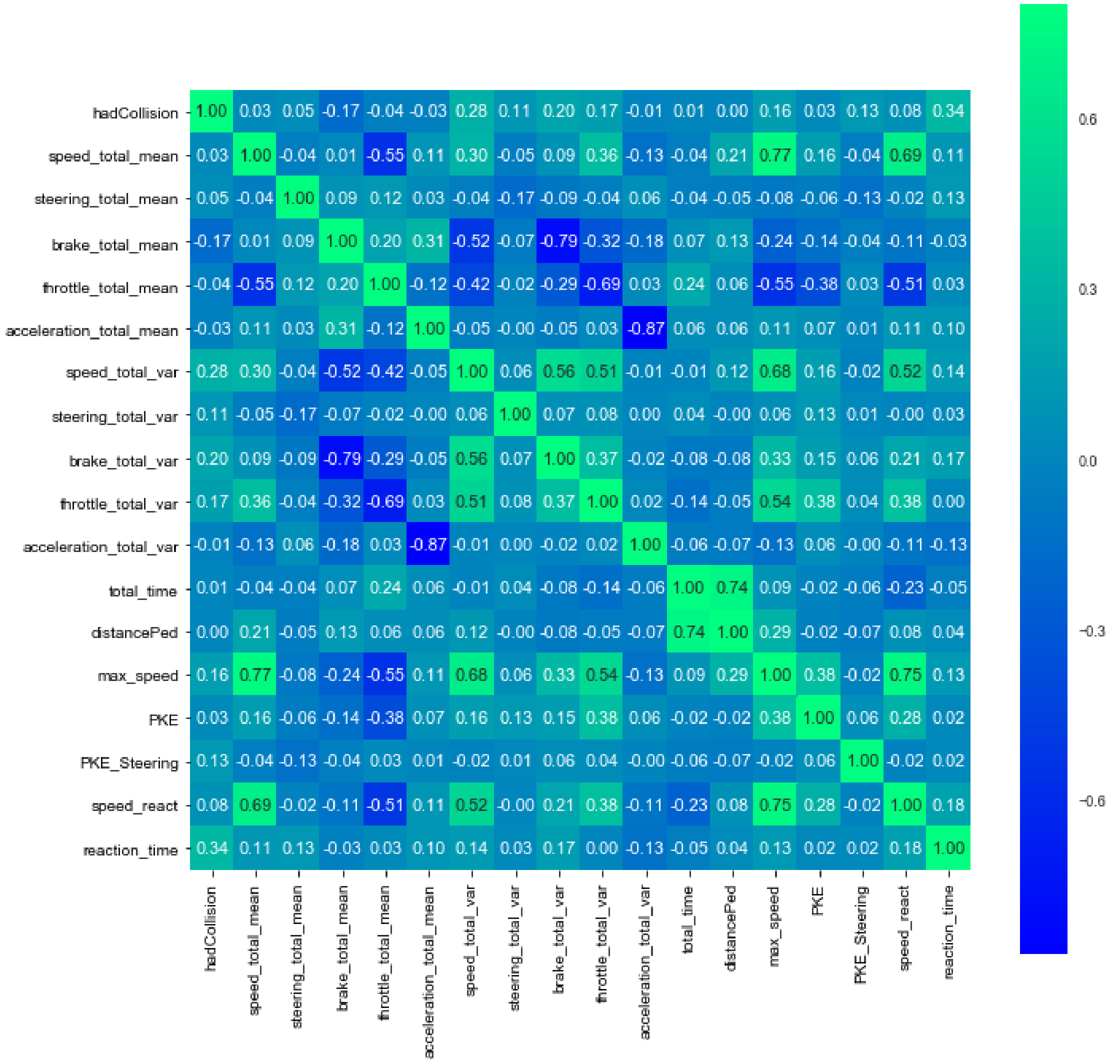
```
Out[93]: (559, 18)
```

```
In [94]: reactionPressBrake.shape
```

```
Out[94]: (559, 5)
```

```
In [95]: def plot_corr(df, size=11):
             """
             Function plots a graphical correlation matrix for each pair of columns in the dat

             Input:
                 df: pandas DataFrame
                 size: vertical and horizontal size of the plot

             Displays:
                 matrix of correlation between columns.  Blue-cyan-yellow-red-darkred => less
                                                         0 ------------------>  1
                                                         Expect a darkred line running from to
             """

             corr = df.corr()    # data frame correlation function
             fig, ax = plt.subplots(figsize=(size, size))
             ax.matshow(corr)   # color code the rectangles by correlation value
             plt.xticks(range(len(corr.columns)), corr.columns)  # draw x tick marks
             plt.yticks(range(len(corr.columns)), corr.columns)  # draw y tick marks
```

```
In [96]: import seaborn as sns

         corrmat = df_suffix.corr()
         f, ax = plt.subplots(figsize=(11,11))
         sns.set(font_scale=0.9)
         sns.heatmap(corrmat, vmax=.8, square=True, annot=True, fmt='.2f', cmap="winter")
         plt.show()
```



```
In [82]: df_suffix.corr()
```

```
Out[82]:                          hadCollision  speed_total_mean  steering_total_mean  \
         hadCollision                 1.000000          0.003023             0.037162
         speed_total_mean             0.003023          1.000000            -0.073901
         steering_total_mean          0.037162         -0.073901             1.000000
         brake_total_mean            -0.103735          0.057452             0.073249
         throttle_total_mean          0.000472         -0.691672             0.129062
         acceleration_total_mean      0.011156          0.320063             0.052155
         speed_total_var              0.212230          0.236173            -0.036002
         steering_total_var           0.147238          0.001821            -0.178203
         brake_total_var              0.143651          0.038165            -0.075838
         throttle_total_var           0.113680          0.382434            -0.047182
         acceleration_total_var      -0.025400         -0.258142            -0.022899
         total_time                   0.031473         -0.056937            -0.022556
```

```
distancePed                  0.032134          0.163800            -0.071290
max_speed                    0.097091          0.821473            -0.080720
PKE                          0.027530          0.153890            -0.038472
PKE_Steering                 0.107471          0.007824            -0.206095
speed_react                  0.077155          0.690015            -0.016004
reaction_time                0.335029          0.113742             0.126902

                         brake_total_mean  throttle_total_mean  \
hadCollision                    -0.103735             0.000472   
speed_total_mean                 0.057452            -0.691672   
steering_total_mean              0.073249             0.129062   
brake_total_mean                 1.000000             0.083151   
throttle_total_mean              0.083151             1.000000   
acceleration_total_mean          0.111709            -0.262172   
speed_total_var                 -0.464997            -0.339819   
steering_total_var              -0.039894            -0.039581   
brake_total_var                 -0.731555            -0.197457   
throttle_total_var              -0.239204            -0.620655   
acceleration_total_var          -0.085719             0.160891   
total_time                       0.064027             0.222875   
distancePed                      0.092040             0.040787   
max_speed                       -0.148294            -0.652242   
PKE                             -0.144054            -0.354183   
PKE_Steering                    -0.034307            -0.026244   
speed_react                     -0.113997            -0.509799   
reaction_time                   -0.032702             0.034894   

                         acceleration_total_mean  speed_total_var  \
hadCollision                            0.011156         0.212230   
speed_total_mean                        0.320063         0.236173   
steering_total_mean                     0.052155        -0.036002   
brake_total_mean                        0.111709        -0.464997   
throttle_total_mean                    -0.262172        -0.339819   
acceleration_total_mean                 1.000000         0.116663   
speed_total_var                         0.116663         1.000000   
steering_total_var                      0.044029         0.088946   
brake_total_var                         0.047181         0.556514   
throttle_total_var                      0.128485         0.490132   
acceleration_total_var                 -0.932188        -0.118702   
total_time                              0.084147         0.012359   
distancePed                             0.101619         0.165060   
max_speed                               0.404565         0.602346   
PKE                                     0.135396         0.215235   
PKE_Steering                            0.006508        -0.006044   
speed_react                             0.113250         0.523054   
reaction_time                           0.101548         0.143505   

                         steering_total_var  brake_total_var  \
hadCollision                       0.147238         0.143651
speed_total_mean                   0.001821         0.038165
steering_total_mean               -0.178203        -0.075838
brake_total_mean                  -0.039894        -0.731555
throttle_total_mean               -0.039581        -0.197457
acceleration_total_mean            0.044029         0.047181
speed_total_var                    0.088946         0.556514
steering_total_var                 1.000000         0.045281
brake_total_var                    0.045281         1.000000
throttle_total_var                 0.105340         0.318190
acceleration_total_var            -0.036567        -0.072101
total_time                         0.053497        -0.075732
distancePed                        0.071798        -0.049071
max_speed                          0.097545         0.267796
PKE                                0.133867         0.171541
PKE_Steering                      -0.011529         0.050653
speed_react                       -0.003533         0.207386
reaction_time                      0.025953         0.165380

                         throttle_total_var  acceleration_total_var  \
hadCollision                       0.113680               -0.025400
speed_total_mean                   0.382434               -0.258142
steering_total_mean               -0.047182               -0.022899
brake_total_mean                  -0.239204               -0.085719
throttle_total_mean               -0.620655                0.160891
acceleration_total_mean            0.128485               -0.932188
speed_total_var                    0.490132               -0.118702
steering_total_var                 0.105340               -0.036567
brake_total_var                    0.318190               -0.072101
throttle_total_var                 1.000000               -0.082509
acceleration_total_var            -0.082509                1.000000
total_time                        -0.115168               -0.089473
distancePed                       -0.022477               -0.101559
max_speed                          0.533691               -0.339401
PKE                                0.374110               -0.055015
PKE_Steering                       0.051558               -0.002641
speed_react                        0.384503               -0.105586
reaction_time                      0.004183               -0.130727

                         total_time  distancePed  max_speed       PKE  \
hadCollision               0.031473     0.032134   0.097091  0.027530
speed_total_mean          -0.056937     0.163800   0.821473  0.153890
steering_total_mean       -0.022556    -0.071290  -0.080720 -0.038472
brake_total_mean           0.064027     0.092040  -0.148294 -0.144054
throttle_total_mean        0.222875     0.040787  -0.652242 -0.354183
acceleration_total_mean    0.084147     0.101619   0.404565  0.135396
speed_total_var            0.012359     0.165060   0.602346  0.215235
steering_total_var         0.053497     0.071798   0.097545  0.133867
```

```
brake_total_var          -0.075732  -0.049071  0.267796  0.171541
throttle_total_var       -0.115168  -0.022477  0.533691  0.374110
acceleration_total_var   -0.089473  -0.101559 -0.339401 -0.055015
total_time                1.000000   0.491870  0.063446 -0.118974
distancePed               0.491870   1.000000  0.277116  0.019337
max_speed                 0.063446   0.277116  1.000000  0.363375
PKE                      -0.118974   0.019337  0.363375  1.000000
PKE_Steering             -0.037848  -0.049021  0.001949  0.079838
speed_react              -0.229716   0.075097  0.749418  0.280204
reaction_time            -0.047035   0.038478  0.128913  0.019924

                         PKE_Steering  speed_react  reaction_time
hadCollision                 0.107471     0.077155       0.335029
speed_total_mean             0.007824     0.690015       0.113742
steering_total_mean         -0.206095    -0.016004       0.126902
brake_total_mean            -0.034307    -0.113997      -0.032702
throttle_total_mean         -0.026244    -0.509799       0.034894
acceleration_total_mean      0.006508     0.113250       0.101548
speed_total_var             -0.006044     0.523054       0.143505
steering_total_var          -0.011529    -0.003533       0.025953
brake_total_var              0.050653     0.207386       0.165380
throttle_total_var           0.051558     0.384503       0.004183
acceleration_total_var      -0.002641    -0.105586      -0.130727
total_time                  -0.037848    -0.229716      -0.047035
distancePed                 -0.049021     0.075097       0.038478
max_speed                    0.001949     0.749418       0.128913
PKE                          0.079838     0.280204       0.019924
PKE_Steering                 1.000000    -0.021700       0.024101
speed_react                 -0.021700     1.000000       0.183851
reaction_time                0.024101     0.183851       1.000000
```

8.1 Mold Data

8.1.1 Data Types

Inspect data types to see if there are any issues. Data should be numeric.

```
In [97]: df = df_suffix
         #hadCollision_map = {True : 1, False : 0}
         df['hadCollision'] = np.where(df['hadCollision'] == 0, 0, 1)
```

```
In [98]: cols_at_end = ['hadCollision']
         df = df[[c for c in df if c not in cols_at_end]
                 + [c for c in cols_at_end if c in df]]
         df.head(5)
```

```
Out[98]:    speed_total_mean  steering_total_mean  brake_total_mean  \
         0          5.919151             0.503649          0.965743
         1          7.580378             0.499771          0.891302
```

```
2            9.474048             0.494557          0.952182
3           11.669419             0.500661          0.891913
4           12.187044             0.499769          0.861132

   throttle_total_mean  acceleration_total_mean  speed_total_var  \
0             0.820576                 0.030731        13.796202
1             0.878839                -0.026652        31.451253
2             0.781126                 0.006292        53.873833
3             0.522365                 0.008028        47.209285
4             0.558120                 0.001881        42.031423

   steering_total_var  brake_total_var  throttle_total_var  \
0            0.000655         0.014468            0.028719
1            0.000345         0.058767            0.010391
2            0.001231         0.022506            0.045416
3            0.000396         0.055982            0.112551
4            0.000430         0.102442            0.079023

   acceleration_total_var  total_time  distancePed  max_speed       PKE  \
0                0.039370   15.405173    89.992450  11.669766  1.932290
1                0.063480   11.412381    85.063860  13.499710  0.878493
2                0.106281  102.356492   789.212800  25.851397  2.857169
3                0.159198    7.505478    88.011610  20.055070  2.969647
4                0.158822    8.681609   105.973686  19.697004  4.033468

   PKE_Steering  speed_react  reaction_time  hadCollision
0     -0.000150     7.754880       1.048791             0
1      0.000274    13.472353       2.106615             0
2      0.000108    25.585112       0.079211             1
3     -0.000258    19.412086       1.161592             0
4      0.000066    18.461056       1.275896             0
```

8.1.2 Backup merge data

```
In [99]: #df.to_csv('mergeData_v5.csv', index=False)
```

8.1.3 Distribution

```
In [86]: num_obs = len(df)
         num_true = len(df.loc[df['hadCollision'] == 1])
         num_false = len(df.loc[df['hadCollision'] == 0])
         print("Number of True cases:  {0} ({1:2.2f}%)".format(num_true, (num_true/num_obs) * 1
         print("Number of False cases: {0} ({1:2.2f}%)".format(num_false, (num_false/num_obs) *
```

```
Number of True cases:  61 (9.31%)
Number of False cases: 594 (90.69%)
```

8.1.4 Spliting the data

70% for training, 30% for testing

```
In [87]: from sklearn.model_selection import train_test_split

         predicted_column = ['hadCollision']

         feature_col_names = [c for c in df if c not in predicted_column]
         predicted_class_names = predicted_column

         X = df[feature_col_names].values     # predictor feature columns (10 X m)
         y = df[predicted_class_names].values # predicted class (1=true, 0=false) column (1 X
         split_test_size = 0.30

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split_test_size,
                                     # test_size = 0.3 is 30%, 42 is the answer to everything
```

We check to ensure we have the the desired 70% train, 30% test split of the data

```
In [88]: print("{0:0.2f}% in training set".format((len(X_train)/len(df.index)) * 100))
         print("{0:0.2f}% in test set".format((len(X_test)/len(df.index)) * 100))
```

```
69.92% in training set
30.08% in test set
```

Verifying predicted value was split correctly

```
In [89]: print("Original True  : {0} ({1:0.2f}%)".format(len(df.loc[df['hadCollision'] == 1]),
         print("Original False : {0} ({1:0.2f}%)".format(len(df.loc[df['hadCollision'] == 0]),
         print("")
         print("Training True  : {0} ({1:0.2f}%)".format(len(y_train[y_train[:] == 1]), (len(y_
         print("Training False : {0} ({1:0.2f}%)".format(len(y_train[y_train[:] == 0]), (len(y_
         print("")
         print("Test True      : {0} ({1:0.2f}%)".format(len(y_test[y_test[:] == 1]), (len(y_te
         print("Test False     : {0} ({1:0.2f}%)".format(len(y_test[y_test[:] == 0]), (len(y_te
```

```
Original True  : 61 (9.31%)
Original False : 594 (90.69%)

Training True  : 48 (10.48%)
Training False : 410 (89.52%)

Test True      : 13 (6.60%)
Test False     : 184 (93.40%)
```

8.1.5 Post-split Data Preparation

```
In [90]: df.head()
```

```
Out[90]:    speed_total_mean  steering_total_mean  brake_total_mean  \
         0          5.919151             0.503649          0.965743
         1          7.580378             0.499771          0.891302
         2          9.474048             0.494557          0.952182
         3          3.398595             0.523305          0.960227
         4         11.669419             0.500661          0.891913

            throttle_total_mean  acceleration_total_mean  speed_total_var  \
         0             0.820576                 0.030731        13.796202
         1             0.878839                -0.026652        31.451253
         2             0.781126                 0.006292        53.873833
         3             0.858795                 0.027363         5.036717
         4             0.522365                 0.008028        47.209285

            steering_total_var  brake_total_var  throttle_total_var  \
         0            0.000655         0.014468            0.028719
         1            0.000345         0.058767            0.010391
         2            0.001231         0.022506            0.045416
         3            0.000101         0.018643            0.075860
         4            0.000396         0.055982            0.112551

            acceleration_total_var  total_time  distancePed  max_speed       PKE  \
         0                0.039370   15.405173    89.992450  11.669766  1.932290
         1                0.063480   11.412381    85.063860  13.499710  0.878493
         2                0.106281  102.356492   789.212800  25.851397  2.857169
         3                0.027735   14.315100    47.977978  10.266865  3.659420
         4                0.159198    7.505478    88.011610  20.055070  2.969647

            PKE_Steering  speed_react  reaction_time  hadCollision
         0     -0.000150     7.754880       1.048791             0
         1      0.000274    13.472353       2.106615             0
         2      0.000108    25.585112       0.079211             1
         3     -0.000627          NaN            NaN             0
         4     -0.000258    19.412086       1.161592             0
```

9 Training Initial Algorithm

9.1 Naive Bayes

```
In [91]: from sklearn.naive_bayes import GaussianNB

         # create Gaussian Naive Bayes model object and train it with the data
         nb_model = GaussianNB()
```

```
 nb_model.fit(X_train, y_train.ravel())
```

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

    ValueError                                Traceback (most recent call last)

    <ipython-input-91-9ac5589a2207> in <module>()
      4 nb_model = GaussianNB()
      5
----> 6 nb_model.fit(X_train, y_train.ravel())


    /anaconda3/lib/python3.6/site-packages/sklearn/naive_bayes.py in fit(self, X, y, sampl
    181             Returns self.
    182         """
--> 183         X, y = check_X_y(X, y)
    184         return self._partial_fit(X, y, np.unique(y), _refit=True,
    185                                  sample_weight=sample_weight)


    /anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py in check_X_y(X, y, a
    571     X = check_array(X, accept_sparse, dtype, order, copy, force_all_finite,
    572                     ensure_2d, allow_nd, ensure_min_samples,
--> 573                     ensure_min_features, warn_on_dtype, estimator)
    574     if multi_output:
    575         y = check_array(y, 'csr', force_all_finite=True, ensure_2d=False,


    /anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py in check_array(arra
    451                              % (array.ndim, estimator_name))
    452         if force_all_finite:
--> 453             _assert_all_finite(array)
    454
    455     shape_repr = _shape_repr(array.shape)


    /anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py in _assert_all_fini
     42             and not np.isfinite(X).all()):
     43         raise ValueError("Input contains NaN, infinity"
---> 44                          " or a value too large for %r." % X.dtype)
     45
     46


    ValueError: Input contains NaN, infinity or a value too large for dtype('float64').
```

9.1.1 Performance on Training Data

```
In [150]: # predict values using the training data
          nb_predict_train = nb_model.predict(X_train)

          # import the performance metrics library
          from sklearn import metrics

          # Accuracy
          print("Accuracy: {0:.4f}".format(metrics.accuracy_score(y_train, nb_predict_train)))
          print()
```

```
Accuracy: 0.6004
```

9.1.2 Performance on Testing Data

```
In [151]: # predict values using the testing data
          nb_predict_test = nb_model.predict(X_test)

          from sklearn import metrics

          # training metrics
          print("nb_predict_test", nb_predict_test)
          #print ("y_test", y_test)
          print("Accuracy: {0:.4f}".format(metrics.accuracy_score(y_test, nb_predict_test)))
```

```
nb_predict_test [0 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0
 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1
 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 1
 1 1 0 1 1 1 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 0 0 0 0
 1 1 0 0 1 0 0 1 1 0 1 0 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 1 1
 0 0 0 1 0 1 0 0 0 0 0 1]
Accuracy: 0.6091
```

Metrics

```
In [152]: print("Confusion Matrix")
          print("{0}".format(metrics.confusion_matrix(y_test, nb_predict_test)))
          print("")

          print("Classification Report")
          print(metrics.classification_report(y_test, nb_predict_test))
```

```
Confusion Matrix
[[112  72]
 [  5   8]]
```

```
Classification Report
             precision    recall  f1-score   support

          0       0.96      0.61      0.74       184
          1       0.10      0.62      0.17        13

avg / total       0.90      0.61      0.71       197
```

9.2 Random Forest

```
In [153]: from sklearn.ensemble import RandomForestClassifier
          rf_model = RandomForestClassifier(random_state=42, n_estimators=10)      # Create ra
          rf_model.fit(X_train, y_train.ravel())
```

```
Out[153]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                      oob_score=False, random_state=42, verbose=0, warm_start=False)
```

9.2.1 Predict Training Data

```
In [154]: rf_predict_train = rf_model.predict(X_train)
          # training metrics
          print("Accuracy: {0:.4f}".format(metrics.accuracy_score(y_train, rf_predict_train)))
```

```
Accuracy: 0.9869
```

9.2.2 Predict Test Data

```
In [155]: rf_predict_test = rf_model.predict(X_test)

          # training metrics
          print("Accuracy: {0:.4f}".format(metrics.accuracy_score(y_test, rf_predict_test)))
```

```
Accuracy: 0.9289
```

```
In [156]: print(metrics.confusion_matrix(y_test, rf_predict_test) )
          print("")
          print("Classification Report")
          print(metrics.classification_report(y_test, rf_predict_test))

          ## TN  FP
          ## FN  TP
```

```
[[183   1]
 [ 13   0]]

Classification Report
             precision    recall  f1-score   support

          0       0.93      0.99      0.96       184
          1       0.00      0.00      0.00        13

avg / total       0.87      0.93      0.90       197
```

9.3 Logistic Regression

```
In [157]: from sklearn.linear_model import LogisticRegression

          #lr_model =LogisticRegression(C=0.7, random_state=42, solver='liblinear', max_iter=1
          lr_model =LogisticRegression(C=0.7, random_state=42)
          lr_model.fit(X_train, y_train.ravel())
          lr_predict_test = lr_model.predict(X_test)

          # training metrics
          print("Accuracy: {0:.4f}".format(metrics.accuracy_score(y_test, lr_predict_test)))
          print(metrics.confusion_matrix(y_test, lr_predict_test) )
          print("")
          print("Classification Report")
          print(metrics.classification_report(y_test, lr_predict_test))
```

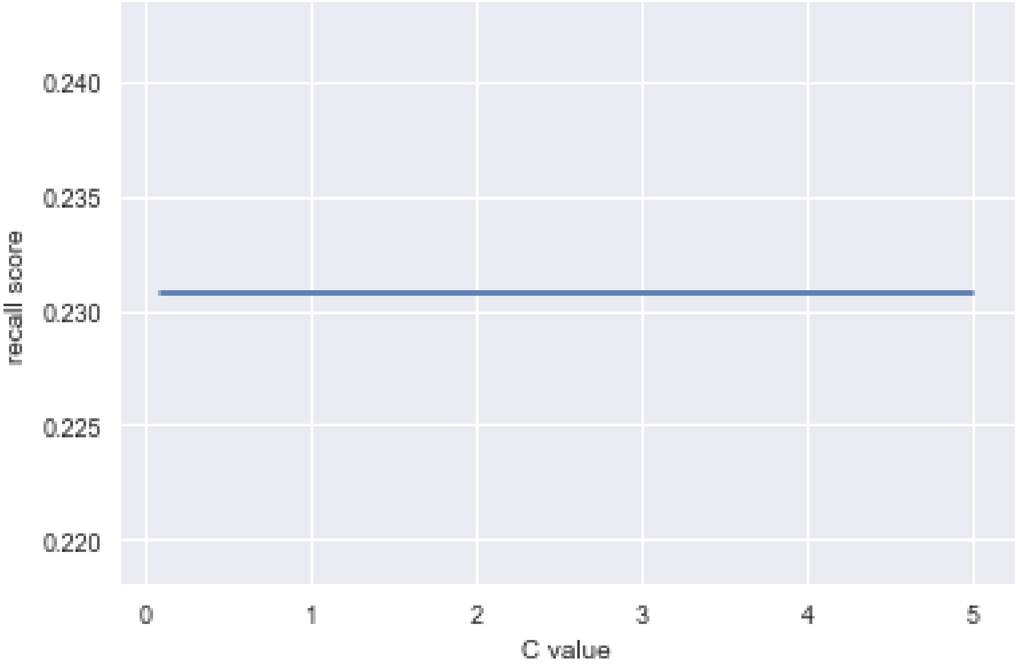
```
Accuracy: 0.9442
[[183   1]
 [ 10   3]]

Classification Report
             precision    recall  f1-score   support

          0       0.95      0.99      0.97       184
          1       0.75      0.23      0.35        13

avg / total       0.94      0.94      0.93       197
```

```
In [158]: C_start = 0.1
          C_end = 5
          C_inc = 0.1

          C_values, recall_scores = [], []
```

```
C_val = C_start
best_recall_score = 0
while (C_val < C_end):
    C_values.append(C_val)
    lr_model_loop = LogisticRegression(C=C_val, random_state=42, solver='liblinear')
    lr_model_loop.fit(X_train, y_train.ravel())
    lr_predict_loop_test = lr_model_loop.predict(X_test)
    recall_score = metrics.recall_score(y_test, lr_predict_loop_test)
    recall_scores.append(recall_score)
    if (recall_score > best_recall_score):
        best_recall_score = recall_score
        best_lr_predict_test = lr_predict_loop_test

    C_val = C_val + C_inc

best_score_C_val = C_values[recall_scores.index(best_recall_score)]
print("1st max value of {0:.3f} occured at C={1:.3f}".format(best_recall_score, best_

%matplotlib inline
plt.plot(C_values, recall_scores, "-")
plt.xlabel("C value")
plt.ylabel("recall score")
```

```
1st max value of 0.231 occured at C=0.100
```

Out[158]: Text(0,0.5,'recall score')

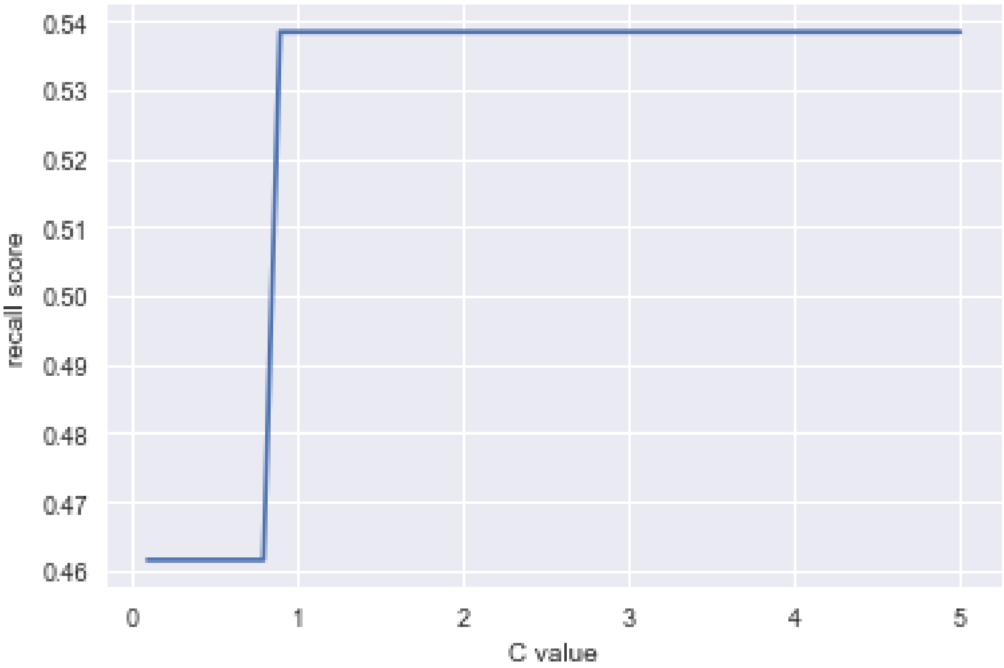


9.3.1 Logisitic regression with class_weight=’balanced’

```
In [159]: C_start = 0.1
          C_end = 5
          C_inc = 0.1

          C_values, recall_scores = [], []

          C_val = C_start
          best_recall_score = 0
          while (C_val < C_end):
              C_values.append(C_val)
              lr_model_loop = LogisticRegression(C=C_val, class_weight="balanced", random_state
              lr_model_loop.fit(X_train, y_train.ravel())
              lr_predict_loop_test = lr_model_loop.predict(X_test)
              recall_score = metrics.recall_score(y_test, lr_predict_loop_test)
              recall_scores.append(recall_score)
              if (recall_score > best_recall_score):
                  best_recall_score = recall_score
                  best_lr_predict_test = lr_predict_loop_test

              C_val = C_val + C_inc

          best_score_C_val = C_values[recall_scores.index(best_recall_score)]
          print("1st max value of {0:.3f} occured at C={1:.3f}".format(best_recall_score, best_

          %matplotlib inline
          plt.plot(C_values, recall_scores, "-")
          plt.xlabel("C value")
          plt.ylabel("recall score")
```

```
1st max value of 0.538 occured at C=0.900
```

```
Out[159]: Text(0,0.5,'recall score')
```



```
In [160]: from sklearn.linear_model import LogisticRegression
          lr_model =LogisticRegression( class_weight="balanced", C=best_score_C_val, random_sta
          lr_model.fit(X_train, y_train.ravel())
          lr_predict_test = lr_model.predict(X_test)

          # training metrics
          print("Accuracy: {0:.4f}".format(metrics.accuracy_score(y_test, lr_predict_test)))
          print(metrics.confusion_matrix(y_test, lr_predict_test) )
          print("")
          print("Classification Report")
          print(metrics.classification_report(y_test, lr_predict_test))
          print(metrics.recall_score(y_test, lr_predict_test))
```

```
Accuracy: 0.6853
[[128  56]
 [  6   7]]

Classification Report
             precision    recall  f1-score   support

          0       0.96      0.70      0.81       184
          1       0.11      0.54      0.18        13

avg / total       0.90      0.69      0.76       197
```

```
0.5384615384615384
```

9.3.2 LogisticRegressionCV

```
In [161]: from sklearn.linear_model import LogisticRegressionCV
          lr_cv_model = LogisticRegressionCV(n_jobs=-1, random_state=42, Cs=3, cv=10, refit=Fal
          lr_cv_model.fit(X_train, y_train.ravel())
```

```
Out[161]: LogisticRegressionCV(Cs=3, class_weight='balanced', cv=10, dual=False,
                     fit_intercept=True, intercept_scaling=1.0, max_iter=500,
                     multi_class='ovr', n_jobs=-1, penalty='l2', random_state=42,
                     refit=False, scoring=None, solver='lbfgs', tol=0.0001,
                     verbose=0)
```

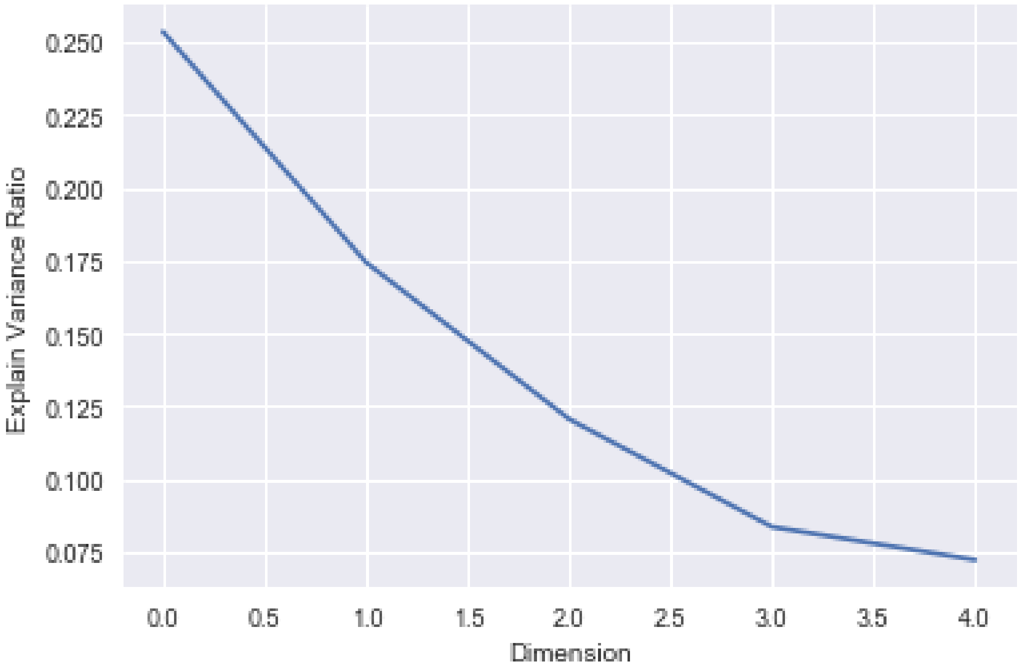
9.3.3 Predict on Test data

```
In [162]: lr_cv_predict_test = lr_cv_model.predict(X_test)

          # training metrics
          print("Accuracy: {0:.4f}".format(metrics.accuracy_score(y_test, lr_cv_predict_test))
          print(metrics.confusion_matrix(y_test, lr_cv_predict_test) )
          print("")
          print("Classification Report")
          print(metrics.classification_report(y_test, lr_cv_predict_test))
```

```
Accuracy: 0.8071
[[152  32]
 [  6   7]]

Classification Report
             precision    recall  f1-score   support

          0       0.96      0.83      0.89       184
          1       0.18      0.54      0.27        13

avg / total       0.91      0.81      0.85       197
```

9.3.4 Linear SVC

```
In [163]: from sklearn.model_selection import train_test_split

          data = df.copy()
          X = data.drop('hadCollision', axis=1)
          Y = data['hadCollision']

          from sklearn import preprocessing
          X = preprocessing.scale(X)


          X_train, x_test, Y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state

In [164]: from sklearn.svm import LinearSVC

          clf_svc = LinearSVC(penalty='l1', dual=False, tol=1e-3)
          clf_svc.fit(X_train, Y_train)

Out[164]: LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,
               intercept_scaling=1, loss='squared_hinge', max_iter=1000,
               multi_class='ovr', penalty='l1', random_state=None, tol=0.001,
               verbose=0)

In [165]: print(clf_svc.score(x_test, y_test))

0.916030534351145


In [166]: from sklearn.decomposition import PCA

          pca = PCA(n_components=5, whiten=True)
          X_reduced = pca.fit_transform(X)

In [167]: pca.explained_variance_

Out[167]: array([6.09179704, 4.19158511, 2.90060492, 2.01183001, 1.73962064])

In [168]: pca.explained_variance_ratio_

Out[168]: array([0.25343736, 0.17438274, 0.12067402, 0.08369827, 0.07237353])

In [169]: plt.plot(pca.explained_variance_ratio_)
          plt.xlabel('Dimension')
          plt.ylabel('Explain Variance Ratio')
          plt.show()
          #Scree plot
```



```
In [170]: X_train, x_test, Y_train, y_test = train_test_split(X_reduced, Y, test_size=0.2, ranc
          clf_svc = LinearSVC(penalty='l1', dual=False, tol=1e-3)
          clf_svc.fit(X_train, Y_train)

Out[170]: LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,
               intercept_scaling=1, loss='squared_hinge', max_iter=1000,
               multi_class='ovr', penalty='l1', random_state=None, tol=0.001,
               verbose=0)

In [171]: print(clf_svc.score(x_test, y_test))

0.9083969465648855
```

Cluster from scenarios

```
In [172]: df_scenario_cluster.head()

Out[172]:    scenario  hadCollision  speed_total_mean  steering_total_mean  \
          0       1.0      0.000000          5.919151             0.503649
          1       1.0      0.000000          7.580378             0.499771
          2       1.0      0.008366          9.474048             0.494557
          3       2.0      0.000000          3.398595             0.523305
          4       2.0      0.000000         11.669419             0.500661
```

```
   brake_total_mean  throttle_total_mean  acceleration_total_mean  \
0          0.965743             0.820576                 0.030731   
1          0.891302             0.878839                -0.026652   
2          0.952182             0.781126                 0.006292   
3          0.960227             0.858795                 0.027363   
4          0.891913             0.522365                 0.008028   

   speed_total_var  steering_total_var  brake_total_var  \
0        13.796202            0.000655         0.014468   
1        31.451253            0.000345         0.058767   
2        53.873833            0.001231         0.022506   
3         5.036717            0.000101         0.018643   
4        47.209285            0.000396         0.055982   

            ...            steering_when_running_var  brake_when_running_var  \
0           ...                             0.000174                0.035741   
1           ...                             0.000151                0.078618   
2           ...                             0.001915                0.121984   
3           ...                             0.000048                0.000000   
4           ...                             0.000429                0.075733   

   throttle_when_running_var  acceleration_when_running_var  \
0                   0.039745                       0.083699   
1                   0.000856                       0.087210   
2                   0.035585                       0.869046   
3                   0.044336                       0.023489   
4                   0.087892                       0.212602   

   speed_when_running_mean  steering_when_running_mean  \
0                 4.593110                    0.476982   
1                 5.276907                    0.508684   
2                 6.933534                    0.473303   
3                 6.170808                    0.499042   
4                 9.577210                    0.504876   

   brake_when_running_mean  throttle_when_running_mean  \
0                 0.900460                    0.739390   
1                 0.839408                    0.929774   
2                 0.769213                    0.707442   
3                 1.000000                    0.075305   
4                 0.836788                    0.630319   

   acceleration_when_running_mean  total_time_when_running
0                        0.016767                 5.239119
1                       -0.065371                 7.705014
2                       -0.068940                 7.848389
3                        0.314027                 1.202199
4                       -0.078850                 4.975241
```

```
[5 rows x 26 columns]
```

```
In [173]: from sklearn.model_selection import train_test_split

          data = df_scenario_cluster.copy()
          X = data.drop('scenario', axis=1)
          Y = data['scenario']

          from sklearn import preprocessing
          X = preprocessing.scale(X)


          X_train, x_test, Y_train, y_test = train_test_split(X, Y, test_size=0.2, random_stat
```

```
In [174]: from sklearn.svm import LinearSVC

          clf_svc = LinearSVC(penalty='l1', dual=False, tol=1e-3)
          clf_svc.fit(X_train, Y_train)
```

```
Out[174]: LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,
               intercept_scaling=1, loss='squared_hinge', max_iter=1000,
               multi_class='ovr', penalty='l1', random_state=None, tol=0.001,
               verbose=0)
```

```
In [175]: print(clf_svc.score(x_test, y_test))
```

```
0.4351145038167939
```

```
In [176]: from sklearn.decomposition import PCA

          pca = PCA(n_components=5, whiten=True)
          X_reduced = pca.fit_transform(X)
```

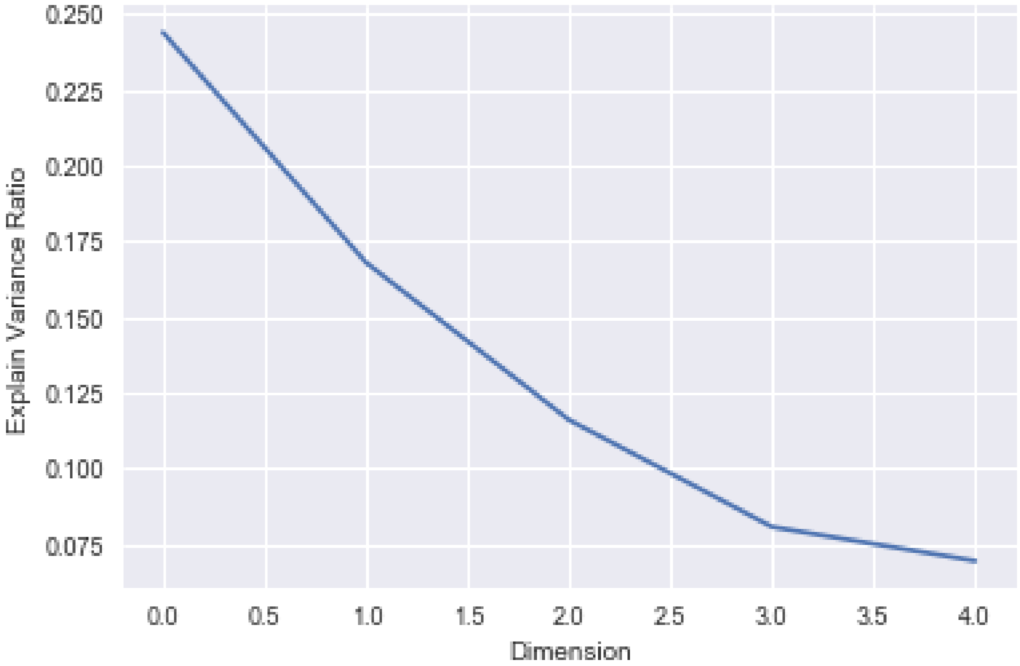
```
In [177]: pca.explained_variance_
```

```
Out[177]: array([6.10683356, 4.20528569, 2.90596682, 2.02242817, 1.74126031])
```

```
In [178]: pca.explained_variance_ratio_
```

```
Out[178]: array([0.24390041, 0.16795462, 0.11606121, 0.08077362, 0.06954408])
```

```
In [179]: plt.plot(pca.explained_variance_ratio_)
          plt.xlabel('Dimension')
          plt.ylabel('Explain Variance Ratio')
          plt.show()
```



```
In [180]: X_train, x_test, Y_train, y_test = train_test_split(X_reduced, Y, test_size=0.2, ranc
          clf_svc = LinearSVC(penalty='l1', dual=False, tol=1e-3)
          clf_svc.fit(X_train, Y_train)

Out[180]: LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,
               intercept_scaling=1, loss='squared_hinge', max_iter=1000,
               multi_class='ovr', penalty='l1', random_state=None, tol=0.001,
               verbose=0)

In [181]: print(clf_svc.score(x_test, y_test))

0.29770992366412213
```

9.3.5 MeanShift

```
In [182]: from sklearn.cluster import MeanShift
          dataFrame = df.copy()

          analyzer = MeanShift(bandwidth=70)
          analyzer.fit(dataFrame)

Out[182]: MeanShift(bandwidth=70, bin_seeding=False, cluster_all=True, min_bin_freq=1,
               n_jobs=1, seeds=None)
```

```
In [183]: from sklearn.cluster import estimate_bandwidth

          estimate_bandwidth(dataFrame)

Out[183]: 80.26694790703168

In [184]: labels = analyzer.labels_

In [185]: np.unique(labels)

Out[185]: array([0, 1, 2, 3, 4, 5, 6])

In [186]: dataFrame['cluster_group'] = np.nan
          data_length = len(dataFrame)
          for i in range(data_length):
              dataFrame.iloc[i, dataFrame.columns.get_loc('cluster_group')] = labels[i]

In [187]: dataFrame.head()

Out[187]:    speed_total_mean  steering_total_mean  brake_total_mean  \
          0          5.919151             0.503649          0.965743   
          1          7.580378             0.499771          0.891302   
          2          9.474048             0.494557          0.952182   
          3          3.398595             0.523305          0.960227   
          4         11.669419             0.500661          0.891913   

             throttle_total_mean  acceleration_total_mean  speed_total_var  \
          0             0.820576                 0.030731        13.796202   
          1             0.878839                -0.026652        31.451253   
          2             0.781126                 0.006292        53.873833   
          3             0.858795                 0.027363         5.036717   
          4             0.522365                 0.008028        47.209285   

             steering_total_var  brake_total_var  throttle_total_var  \
          0            0.000655         0.014468            0.028719   
          1            0.000345         0.058767            0.010391   
          2            0.001231         0.022506            0.045416   
          3            0.000101         0.018643            0.075860   
          4            0.000396         0.055982            0.112551   

             acceleration_total_var      ...        throttle_when_running_var  \
          0                0.039370      ...                         0.039745   
          1                0.063480      ...                         0.000856   
          2                0.106281      ...                         0.035585   
          3                0.027735      ...                         0.044336   
          4                0.159198      ...                         0.087892   

             acceleration_when_running_var  speed_when_running_mean  \
          0                       0.083699                 4.593110   
```

```
1                       0.087210                 5.276907
2                       0.869046                 6.933534
3                       0.023489                 6.170808
4                       0.212602                 9.577210

   steering_when_running_mean  brake_when_running_mean  \
0                    0.476982                 0.900460
1                    0.508684                 0.839408
2                    0.473303                 0.769213
3                    0.499042                 1.000000
4                    0.504876                 0.836788

   throttle_when_running_mean  acceleration_when_running_mean  \
0                    0.739390                        0.016767
1                    0.929774                       -0.065371
2                    0.707442                       -0.068940
3                    0.075305                        0.314027
4                    0.630319                       -0.078850

   total_time_when_running  hadCollision  cluster_group
0                 5.239119             0            0.0
1                 7.705014             0            0.0
2                 7.848389             1            2.0
3                 1.202199             0            0.0
4                 4.975241             0            0.0

[5 rows x 26 columns]
```

```
In [188]: dataFrame.describe()
```

```
Out[188]:        speed_total_mean  steering_total_mean  brake_total_mean  \
          count        655.000000           655.000000        655.000000
          mean           7.987191             0.502204          0.959427
          std            3.740045             0.010688          0.052920
          min            0.015117             0.418696          0.441398
          25%            5.757110             0.497913          0.944744
          50%            7.474806             0.501098          0.975665
          75%            9.599790             0.505841          0.991609
          max           35.358340             0.561057          1.000000

                 throttle_total_mean  acceleration_total_mean  speed_total_var  \
          count           655.000000               655.000000       655.000000
          mean              0.812115                -0.020020        15.369249
          std               0.126159                 0.153846        13.677285
          min               0.199416                -1.779510         0.000251
          25%               0.771546                -0.008199         7.271839
          50%               0.831697                 0.000143        12.569265
          75%               0.883929                 0.008954        19.953399
```

```
max                1.000000                 0.171440       124.624108

       steering_total_var  brake_total_var  throttle_total_var  \
count          655.000000       655.000000          655.000000
mean             0.000927         0.018102            0.032908
std              0.002787         0.027826            0.037133
min              0.000000         0.000000            0.000000
25%              0.000078         0.000966            0.011291
50%              0.000304         0.005730            0.020177
75%              0.000662         0.024807            0.039857
max              0.037609         0.200328            0.226159

       acceleration_total_var      ...        throttle_when_running_var  \
count              655.000000      ...                       655.000000
mean                 0.273928      ...                         0.035769
std                  1.551995      ...                         0.041068
min                  0.000007      ...                         0.000000
25%                  0.021014      ...                         0.009976
50%                  0.037177      ...                         0.021740
75%                  0.074261      ...                         0.044298
max                 19.011509      ...                         0.232309

       acceleration_when_running_var  speed_when_running_mean  \
count                   6.550000e+02               655.000000
mean                    2.562516e-01                 7.291854
std                     1.531710e+00                 4.108181
min                     1.663293e-07                 0.015117
25%                     2.505745e-02                 4.883573
50%                     5.581903e-02                 6.459509
75%                     1.002668e-01                 8.786491
max                     1.901151e+01                36.319434

       steering_when_running_mean  brake_when_running_mean  \
count                  655.000000               655.000000
mean                     0.501457                 0.924892
std                      0.015147                 0.082771
min                      0.385388                 0.441398
25%                      0.496080                 0.887307
50%                      0.500045                 0.953593
75%                      0.505914                 0.985116
max                      0.589599                 1.000000

       throttle_when_running_mean  acceleration_when_running_mean  \
count                  655.000000                      655.000000
mean                     0.808358                       -0.044074
std                      0.143912                        0.155841
min                      0.075305                       -1.779510
25%                      0.757800                       -0.052260
```

```
50%                      0.835322                 -0.025605
75%                      0.896792                 -0.002892
max                      1.000000                  0.314027

       total_time_when_running  hadCollision  cluster_group
count               655.000000    655.000000     655.000000
mean                 10.767179      0.093130       0.247328
std                  31.374280      0.290836       0.748018
min                   0.150901      0.000000       0.000000
25%                   4.152824      0.000000       0.000000
50%                   5.155710      0.000000       0.000000
75%                   6.358851      0.000000       0.000000
max                 344.235341      1.000000       6.000000

[8 rows x 26 columns]
```

```
In [189]: dataFrameCluster = dataFrame.groupby(['cluster_group']).mean()
          dataFrameCluster
```

```
Out[189]:                speed_total_mean  steering_total_mean  brake_total_mean  \
          cluster_group
          0.0                    7.887470             0.502157          0.958623
          1.0                    8.825458             0.503529          0.963621
          2.0                    9.232445             0.499030          0.976546
          3.0                    7.572369             0.502950          0.974084
          4.0                    9.322995             0.508132          0.909123
          5.0                    2.641436             0.499082          0.933273
          6.0                    3.072536             0.497784          0.992410

                         throttle_total_mean  acceleration_total_mean  speed_total_var  \
          cluster_group
          0.0                       0.806904                -0.022963        14.705734
          1.0                       0.818967                 0.003037        21.534679
          2.0                       0.837564                 0.001770        22.508241
          3.0                       0.910601                -0.011341        12.586728
          4.0                       0.895185                -0.014790        16.368998
          5.0                       0.974126                -0.002999         6.484751
          6.0                       0.969904                -0.003625         2.967740

                         steering_total_var  brake_total_var  throttle_total_var  \
          cluster_group
          0.0                      0.000879         0.018488            0.033597
          1.0                      0.001373         0.019832            0.030851
          2.0                      0.001884         0.010716            0.028988
          3.0                      0.000204         0.007492            0.024185
          4.0                      0.000609         0.021577            0.025884
          5.0                      0.000026         0.019491            0.002310
          6.0                      0.000033         0.000902            0.004328
```

```
               acceleration_total_var      ...       brake_when_running_var  \
cluster_group                              ...
0.0                          0.307975      ...                     0.030969
1.0                          0.032106      ...                     0.047379
2.0                          0.027321      ...                     0.049719
3.0                          0.095436      ...                     0.009590
4.0                          0.175735      ...                     0.023818
5.0                          0.150991      ...                     0.021568
6.0                          0.033197      ...                     0.000006

               throttle_when_running_var  acceleration_when_running_var  \
cluster_group
0.0                             0.035905                       0.280664
1.0                             0.034138                       0.080932
2.0                             0.048340                       0.123751
3.0                             0.025079                       0.083352
4.0                             0.023126                       0.171621
5.0                             0.003742                       0.012650
6.0                             0.000267                       0.017874

               speed_when_running_mean  steering_when_running_mean  \
cluster_group
0.0                           7.277498                    0.501614
1.0                           7.647502                    0.500095
2.0                           7.410295                    0.495831
3.0                           6.738427                    0.504081
4.0                           8.424250                    0.512584
5.0                           2.570975                    0.501220
6.0                           2.312557                    0.495784

               brake_when_running_mean  throttle_when_running_mean  \
cluster_group
0.0                           0.926740                    0.809789
1.0                           0.905508                    0.767292
2.0                           0.897335                    0.732501
3.0                           0.966015                    0.935004
4.0                           0.893861                    0.919901
5.0                           0.910892                    0.968243
6.0                           0.999638                    0.995387

               acceleration_when_running_mean  total_time_when_running  \
cluster_group
0.0                                 -0.049503                 5.630962
1.0                                 -0.008089                 6.032106
2.0                                 -0.000749                 6.589248
3.0                                 -0.021705               201.165259
4.0                                 -0.014631                79.213483
```

```
5.0                                    -0.004881              9.074812
6.0                                    -0.003943            344.235341

               hadCollision
cluster_group
0.0                0.077465
1.0                0.181818
2.0                0.272727
3.0                0.000000
4.0                0.166667
5.0                1.000000
6.0                1.000000

[7 rows x 25 columns]
```

```
In [190]: dataFrameCluster['Counts'] = pd.Series(dataFrame.groupby(['cluster_group']).size())
          dataFrameCluster
```

```
Out[190]:                speed_total_mean  steering_total_mean  brake_total_mean  \
          cluster_group
          0.0                    7.887470             0.502157          0.958623
          1.0                    8.825458             0.503529          0.963621
          2.0                    9.232445             0.499030          0.976546
          3.0                    7.572369             0.502950          0.974084
          4.0                    9.322995             0.508132          0.909123
          5.0                    2.641436             0.499082          0.933273
          6.0                    3.072536             0.497784          0.992410

                         throttle_total_mean  acceleration_total_mean  speed_total_var  \
          cluster_group
          0.0                       0.806904                -0.022963        14.705734
          1.0                       0.818967                 0.003037        21.534679
          2.0                       0.837564                 0.001770        22.508241
          3.0                       0.910601                -0.011341        12.586728
          4.0                       0.895185                -0.014790        16.368998
          5.0                       0.974126                -0.002999         6.484751
          6.0                       0.969904                -0.003625         2.967740

                         steering_total_var  brake_total_var  throttle_total_var  \
          cluster_group
          0.0                      0.000879         0.018488            0.033597
          1.0                      0.001373         0.019832            0.030851
          2.0                      0.001884         0.010716            0.028988
          3.0                      0.000204         0.007492            0.024185
          4.0                      0.000609         0.021577            0.025884
          5.0                      0.000026         0.019491            0.002310
          6.0                      0.000033         0.000902            0.004328
```

```
               acceleration_total_var  ...     throttle_when_running_var  \
cluster_group                          ...
0.0                          0.307975  ...                      0.035905
1.0                          0.032106  ...                      0.034138
2.0                          0.027321  ...                      0.048340
3.0                          0.095436  ...                      0.025079
4.0                          0.175735  ...                      0.023126
5.0                          0.150991  ...                      0.003742
6.0                          0.033197  ...                      0.000267

               acceleration_when_running_var  speed_when_running_mean  \
cluster_group
0.0                                 0.280664                 7.277498
1.0                                 0.080932                 7.647502
2.0                                 0.123751                 7.410295
3.0                                 0.083352                 6.738427
4.0                                 0.171621                 8.424250
5.0                                 0.012650                 2.570975
6.0                                 0.017874                 2.312557

               steering_when_running_mean  brake_when_running_mean  \
cluster_group
0.0                              0.501614                 0.926740
1.0                              0.500095                 0.905508
2.0                              0.495831                 0.897335
3.0                              0.504081                 0.966015
4.0                              0.512584                 0.893861
5.0                              0.501220                 0.910892
6.0                              0.495784                 0.999638

               throttle_when_running_mean  acceleration_when_running_mean  \
cluster_group
0.0                              0.809789                       -0.049503
1.0                              0.767292                       -0.008089
2.0                              0.732501                       -0.000749
3.0                              0.935004                       -0.021705
4.0                              0.919901                       -0.014631
5.0                              0.968243                       -0.004881
6.0                              0.995387                       -0.003943

               total_time_when_running  hadCollision  Counts
cluster_group
0.0                           5.630962      0.077465     568
1.0                           6.032106      0.181818      44
2.0                           6.589248      0.272727      22
3.0                         201.165259      0.000000      13
4.0                          79.213483      0.166667       6
5.0                           9.074812      1.000000       1
```

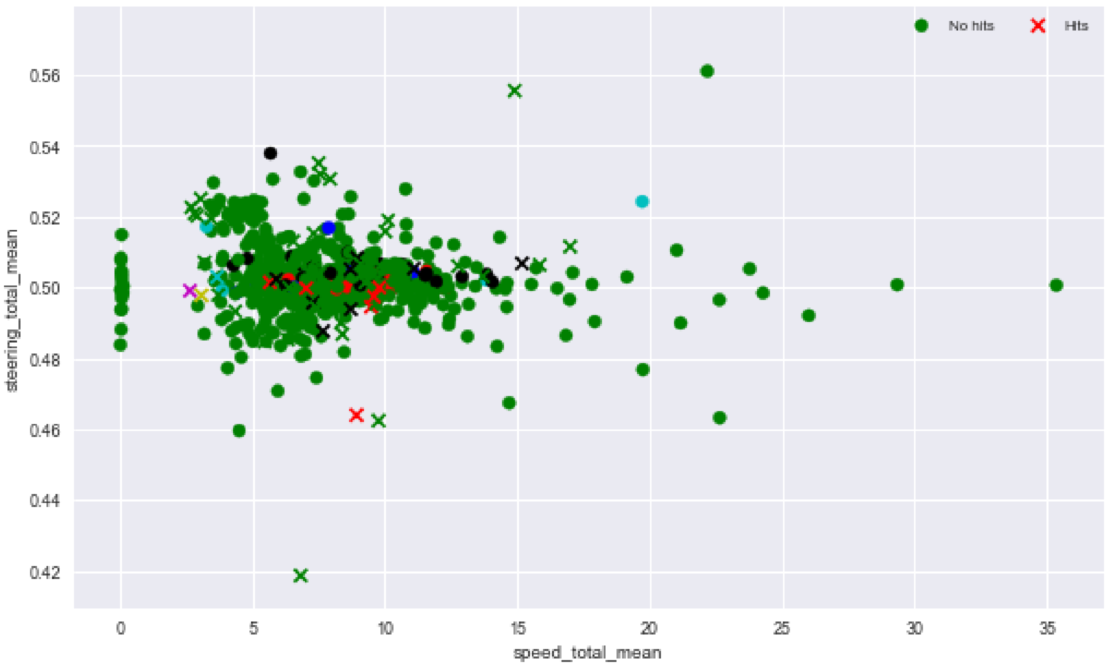
```
6.0                          344.235341      1.000000       1

[7 rows x 26 columns]
```

```
In [191]: dataFrame[dataFrame['cluster_group'] == 2.0].describe()
```

```
Out[191]:        speed_total_mean  steering_total_mean  brake_total_mean  \
          count         22.000000            22.000000         22.000000
          mean           9.232445             0.499030          0.976546
          std            1.608238             0.008093          0.013095
          min            5.653445             0.463986          0.948506
          25%            8.272409             0.499774          0.967847
          50%            9.500332             0.500607          0.978298
          75%           10.183151             0.501842          0.984882
          max           12.279223             0.504581          0.996079

                 throttle_total_mean  acceleration_total_mean  speed_total_var  \
          count            22.000000                22.000000        22.000000
          mean              0.837564                 0.001770        22.508241
          std               0.048628                 0.001965        10.582811
          min               0.737526                -0.001478         6.450247
          25%               0.794283                 0.000632        16.599492
          50%               0.851024                 0.001406        21.486849
          75%               0.867522                 0.002863        26.074040
          max               0.913743                 0.006292        53.873833

                 steering_total_var  brake_total_var  throttle_total_var  \
          count           22.000000        22.000000           22.000000
          mean             0.001884         0.010716            0.028988
          std              0.005150         0.011309            0.020958
          min              0.000217         0.000506            0.005739
          25%              0.000340         0.002784            0.012855
          50%              0.000544         0.006052            0.021298
          75%              0.001216         0.017754            0.043858
          max              0.024657         0.037204            0.083085

                 acceleration_total_var     ...       throttle_when_running_var  \
          count               22.000000     ...                       22.000000
          mean                 0.027321     ...                        0.048340
          std                  0.020993     ...                        0.038391
          min                  0.007523     ...                        0.002528
          25%                  0.015021     ...                        0.028253
          50%                  0.021042     ...                        0.036624
          75%                  0.031946     ...                        0.064602
          max                  0.106281     ...                        0.169520

                 acceleration_when_running_var  speed_when_running_mean  \
          count                      22.000000                22.000000
```

```
mean                         0.123751                7.410295
std                          0.176535                2.723078
min                          0.011597                4.257427
25%                          0.044624                5.495379
50%                          0.081811                6.714656
75%                          0.117960                8.176628
max                          0.869046               14.278999

       steering_when_running_mean  brake_when_running_mean  \
count                   22.000000                22.000000
mean                     0.495831                 0.897335
std                      0.008894                 0.099058
min                      0.470238                 0.611971
25%                      0.493801                 0.890907
50%                      0.497187                 0.926809
75%                      0.501390                 0.960155
max                      0.505552                 1.000000

       throttle_when_running_mean  acceleration_when_running_mean  \
count                   22.000000                       22.000000
mean                     0.732501                       -0.000749
std                      0.157627                        0.055754
min                      0.279519                       -0.068940
25%                      0.651544                       -0.024866
50%                      0.769994                       -0.015546
75%                      0.830914                        0.003633
max                      0.951579                        0.208821

       total_time_when_running  hadCollision  cluster_group
count                22.000000     22.000000           22.0
mean                  6.589248      0.272727            2.0
std                   4.290100      0.455842            0.0
min                   1.889065      0.000000            2.0
25%                   4.458859      0.000000            2.0
50%                   5.486243      0.000000            2.0
75%                   6.962391      0.750000            2.0
max                  21.506701      1.000000            2.0

[8 rows x 26 columns]
```

```
In [230]: plt.figure(figsize=(10, 6))

          #dataFrame = reactionPressBrake[reactionPressBrake.reaction_time.dt.total_seconds()

          LABEL_COLOR_MAP = {0 : 'g',
                             1 : 'k',
                             2 : 'r',
                             3 : 'b',
                             4 : 'c',
                             5 : 'm',
                             6 : 'y'
                             }

          label_color = [LABEL_COLOR_MAP[l] for l in dataFrame['cluster_group']]

          markerMap = np.where(dataFrame['hadCollision'] == 0, 'o', 'x')
          dataFrameNoHits = dataFrame[dataFrame.hadCollision == 0]
          label_color_no_hit = [LABEL_COLOR_MAP[l] for l in dataFrameNoHits['cluster_group']]
          dataFrameHits = dataFrame[dataFrame.hadCollision == 1]
          label_color_hit = [LABEL_COLOR_MAP[l] for l in dataFrameHits['cluster_group']]


          x = 'speed_total_mean'
          y = 'steering_total_mean'
          c = 'cluster_group'

          noHits = plt.scatter(dataFrameNoHits[x], dataFrameNoHits[y], c=label_color_no_hit, ma
          Hits = plt.scatter(dataFrameHits[x], dataFrameHits[y], c=label_color_hit, marker='x')

          plt.xlabel(x)
          plt.ylabel(y)

          plt.legend(( noHits,  Hits),
                     ("No hits","Hits"),
                     scatterpoints=1,
                     ncol=3,
                     fontsize=8)

          plt.grid(True)

          plt.show()
```



```
In [ ]: ## Como tomar los datos de session a partir de acá
```