janakparajuli_api_request_deckgl

April 14, 2020

Janak Parajuli MSc. in Geospatial Technologies (2nd Sem) University of Muenster

Assignment I Floating Car Data Analytics

1 Package loading and basic configurations

[5]:
```
%load_ext autoreload
%autoreload 2

# load dependencies'
import pandas as pd
import geopandas as gpd

from envirocar import TrackAPI, DownloadClient, BboxSelector, ECConfig

# create an initial but optional config and an api client
config = ECConfig()
track_api = TrackAPI(api_client=DownloadClient(config=config))
```

2 Querying enviroCar Tracks

The following cell queries tracks from the enviroCar API. It defines a bbox for the area of Münster (Germany) and requests 50 tracks. The result is a GeoDataFrame, which is a geo-extended Pandas dataframe from the GeoPandas library. It contains all information of the track in a flat dataframe format including a specific geometry column.

[6]:
```
#bbox = BboxSelector([
#    7.501165771484380, # min_x, min longitude
#    51.94807412325402, # min_y, min latitude
#    7.548200988769531, # max_x, max longitude
#    51.97261482608728  # max_y, max latitude
#])
bbox = BboxSelector([
    7.318136,51.802163, 7.928939,52.105665
])
#7.318136,51.802163, 7.928939,52.105665
# issue a query
```

```
track_df = track_api.get_tracks(bbox=bbox, num_results=50) # requesting 50␣
↪tracks inside the bbox
track_df
```

```
[6]:                          id                 time                   geometry  \
    0   5e8baea465b80c5d6b4dbfbf  2020-04-06T20:43:35  POINT (7.65079 51.95400)
    1   5e8baea465b80c5d6b4dbfc1  2020-04-06T20:43:40  POINT (7.65079 51.95412)
    2   5e8baea465b80c5d6b4dbfc2  2020-04-06T20:43:45  POINT (7.65083 51.95435)
    3   5e8baea465b80c5d6b4dbfc3  2020-04-06T20:43:50  POINT (7.65086 51.95463)
    4   5e8baea465b80c5d6b4dbfc4  2020-04-06T20:43:55  POINT (7.65090 51.95480)
    ..                       …                    …                          …
    63  5e08bc785bc8db42896408b7  2019-12-21T11:56:15  POINT (7.64402 51.97021)
    64  5e08bc785bc8db42896408b8  2019-12-21T11:56:20  POINT (7.64402 51.97020)
    65  5e08bc785bc8db42896408b9  2019-12-21T11:56:25  POINT (7.64402 51.97020)
    66  5e08bc785bc8db42896408ba  2019-12-21T11:56:30  POINT (7.64404 51.97018)
    67  5e08bc785bc8db42896408bb  2019-12-21T11:56:36  POINT (7.64404 51.97018)

        GPS Altitude.value GPS Altitude.unit  GPS Bearing.value GPS Bearing.unit  \
    0           100.237808                 m         337.001680              deg
    1           102.772222                 m          11.636667              deg
    2           104.020541                 m           6.089730              deg
    3           103.999999                 m           4.503939              deg
    4           104.000001                 m           7.967200              deg
    ..                 …               …                  …                …
    63          110.000003                 m           0.000000              deg
    64          109.999997                 m           0.000000              deg
    65          109.554884                 m         150.086107              deg
    66          111.000000                 m           0.000000              deg
    67          111.000003                 m           0.000000              deg

        Throttle Position.value Throttle Position.unit  Speed.value  …  \
    0                 16.283688                      %     6.000000  …
    1                 17.920277                      %    14.260606  …
    2                 16.000000                      %    23.999999  …
    3                 16.000000                      %    21.000001  …
    4                 16.000000                      %     3.000000  …
    ..                      …                    …           …  …
    63                15.000000                      %     3.000000  …
    64                16.663317                      %     0.000000  …
    65                15.000000                      %     2.000000  …
    66                15.000000                      %     0.000000  …
    67                      NaN                    NaN          NaN  …

       sensor.constructionYear  sensor.manufacturer track.appVersion  \
    0                     2007                Dodge              NaN
    1                     2007                Dodge              NaN
    2                     2007                Dodge              NaN
```

```
3                   2007              Dodge              NaN
4                   2007              Dodge              NaN
..                   …                 …              …
63                  2007              Dodge              NaN
64                  2007              Dodge              NaN
65                  2007              Dodge              NaN
66                  2007              Dodge              NaN
67                  2007              Dodge              NaN

    track.touVersion O2 Lambda Voltage.value  O2 Lambda Voltage.unit  \
0                NaN                     NaN                     NaN
1                NaN                     NaN                     NaN
2                NaN                     NaN                     NaN
3                NaN                     NaN                     NaN
4                NaN                     NaN                     NaN
..               …                    …                     …
63               NaN                     NaN                     NaN
64               NaN                     NaN                     NaN
65               NaN                     NaN                     NaN
66               NaN                     NaN                     NaN
67               NaN                     NaN                     NaN

   MAF.value  MAF.unit O2 Lambda Voltage ER.value  O2 Lambda Voltage ER.unit
0        NaN       NaN                        NaN                        NaN
1        NaN       NaN                        NaN                        NaN
2        NaN       NaN                        NaN                        NaN
3        NaN       NaN                        NaN                        NaN
4        NaN       NaN                        NaN                        NaN
..       …       …                        …                        …
63       NaN       NaN                        NaN                        NaN
64       NaN       NaN                        NaN                        NaN
65       NaN       NaN                        NaN                        NaN
66       NaN       NaN                        NaN                        NaN
67       NaN       NaN                        NaN                        NaN

[16254 rows x 54 columns]
```

[7]:
```
print(track_df.describe()) #Summary statistics of numeric column
```

```
       GPS Altitude.value  GPS Bearing.value  Throttle Position.value  \
count        16254.000000       15953.000000             14669.000000
mean            91.724343         145.922260                26.973201
std             25.261047         109.631387                18.528991
min             30.999999          -2.304270                10.000000
25%             78.435980          40.707150                16.000000
50%             97.203334         149.100006                21.368312
75%            105.000002         226.992945                27.165468
```

```
max          195.999997        363.849744              89.000003

       Speed.value  GPS PDOP.value  Intake Temperature.value  GPS VDOP.value  \
count  15233.000000    13825.000000              14669.000000    13825.000000
mean      76.564161        1.066603                 11.733752        0.843555
std       43.948642        0.367407                  7.043985        0.307673
min        0.000000        0.800000                  3.000000        0.600000
25%       43.000000        0.900000                  7.000000        0.700000
50%       79.999998        1.000000                  9.000000        0.800000
75%      118.000000        1.100000                 15.000000        0.808105
max      373.333340        9.975758                 37.999999        8.578788

       GPS Speed.value  Intake Pressure.value  Calculated MAF.value  …  \
count     16254.000000           14668.000000          12886.000000  …
mean         74.921288              66.919684             23.172532  …
std          44.784828              32.065002             14.963303  …
min           0.000000              16.000000             -7.269221  …
25%          40.227443              44.544775              8.846632  …
50%          77.546974              65.784226             21.190572  …
75%         117.807371              80.587479             35.552383  …
max         174.567824             255.000000             72.078894  …

         Rpm.value  GPS HDOP.value  GPS Accuracy.value  Engine Load.value  \
count  15233.000000    13825.000000        16254.000000       15233.000000
mean    2186.558264        0.583288            2.764771          45.636294
std      949.642333        0.235511            2.055794          26.821396
min     -859.118241        0.400000            1.000000        -495.792866
25%     1482.735338        0.400000            1.500000          27.022249
50%     2056.895271        0.600000            2.000000          47.058823
75%     3125.371775        0.639722            3.564713          65.840351
max     4530.827519        5.444747           45.526076         553.634987

       track.length  sensor.engineDisplacement  sensor.constructionYear  \
count  16254.000000               16254.000000             16254.000000
mean     114.723543                1741.772917              2008.812477
std       87.325675                 199.288250                 4.299975
min        0.000000                1328.000000              1999.000000
25%       15.404851                1798.000000              2007.000000
50%      161.712887                1798.000000              2007.000000
75%      171.928734                1798.000000              2007.000000
max      233.951996                2461.000000              2019.000000

       O2 Lambda Voltage.value    MAF.value  O2 Lambda Voltage ER.value
count              1046.000000  1783.000000                 1046.000000
mean                  0.605838    24.633573                    1.718217
std                   0.344659    21.500357                    0.302982
min                   0.000000     1.733910                    0.995972
25%                   0.358227    10.591114                    1.476118
```

```
50%                  0.521146    19.779265                       1.809511
75%                  0.836780    31.661124                       1.999969
max                  1.270696   240.804784                       1.999970

[8 rows x 22 columns]
```

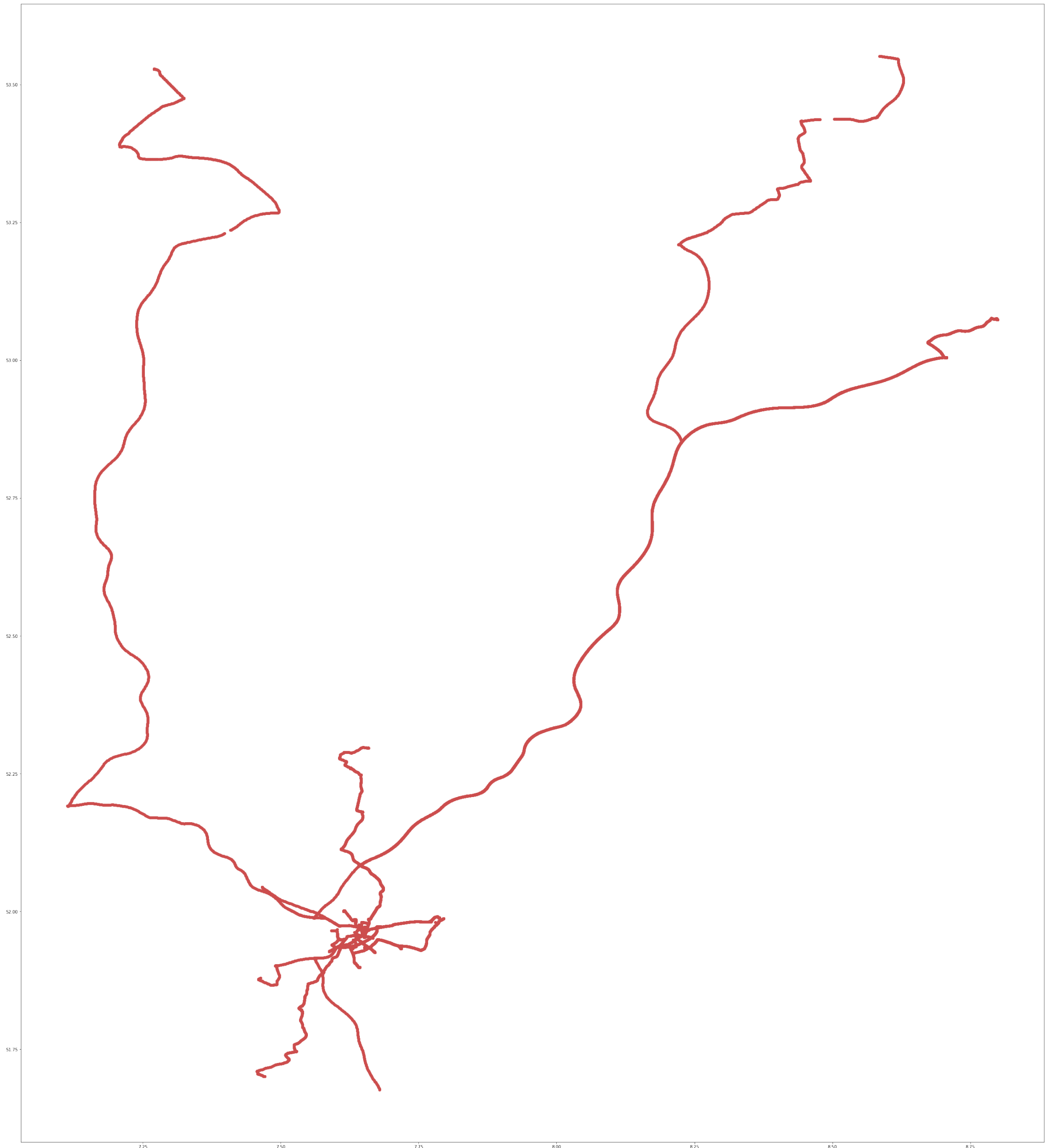
[8]:
```
print(track_df.describe(include=['object']))  #Summary statistic of non-numeric column
```

```
                              id                 time GPS Altitude.unit  \
count                      16254                16254             16254
unique                     16254                16073                 1
top     5e08bc845bc8db428964257a  2020-04-06T10:02:46                 m
freq                           1                    2             16254

       GPS Bearing.unit Throttle Position.unit Speed.unit GPS PDOP.unit  \
count             15953                  14669      15233         13825
unique                1                      1          1             1
top                 deg                      %       km/h     precision
freq              15953                  14669      15233         13825

       Intake Temperature.unit GPS VDOP.unit GPS Speed.unit  … sensor.type  \
count                    14669         13825          16254  …       16254
unique                       1             1              1  …           1
top                          c     precision           km/h  …         car
freq                     14669         13825          16254  …       16254

       sensor.model                 sensor.id sensor.fuelType  \
count         16254                     16254           16254
unique            6                         7               2
top         Caliber  58395f40e4b0a979d45bd61b        gasoline
freq          11128                     11128           14564

       sensor.manufacturer                    track.appVersion  \
count                16254                                 209
unique                   6                                   1
top                  Dodge  Version 1.0.2 (38), 30.11.79 00:00
freq                 11128                                 209

       track.touVersion O2 Lambda Voltage.unit MAF.unit  \
count               209                   1046     1783
unique                1                      1        1
top          2013-10-01                      V      l/s
freq                209                   1046     1783

       O2 Lambda Voltage ER.unit
count                       1046
unique                         1
```

```
top                        ratio
freq                        1046

[4 rows x 31 columns]
```

[59]:
```python
track_df.plot(figsize=(260, 50), color=(0.8,0.3,0.3))
```

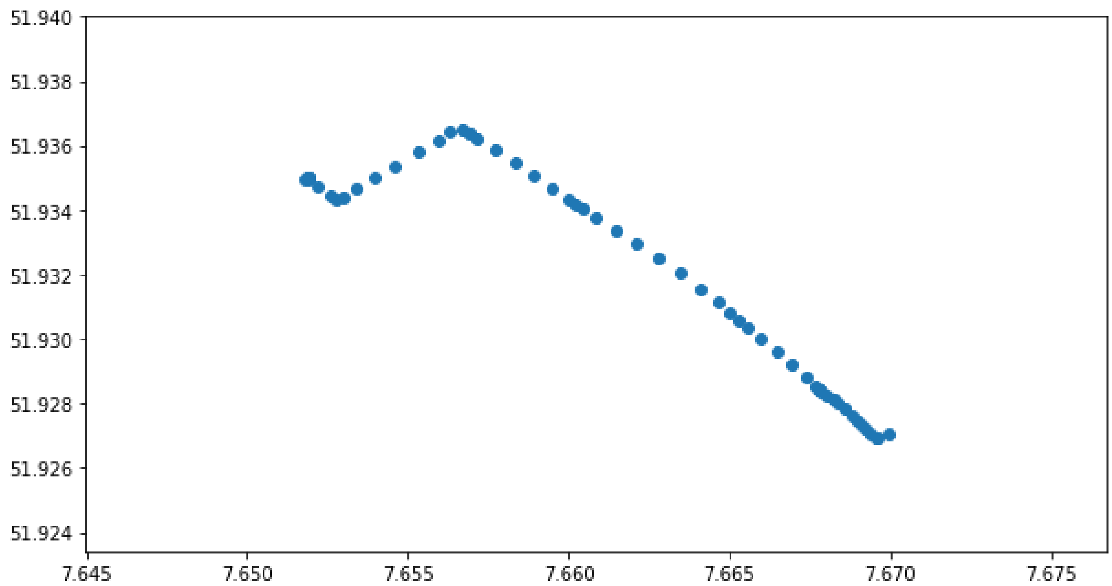
[59]: `<matplotlib.axes._subplots.AxesSubplot at 0x23df3282848>`



3 Inspecting a single Track

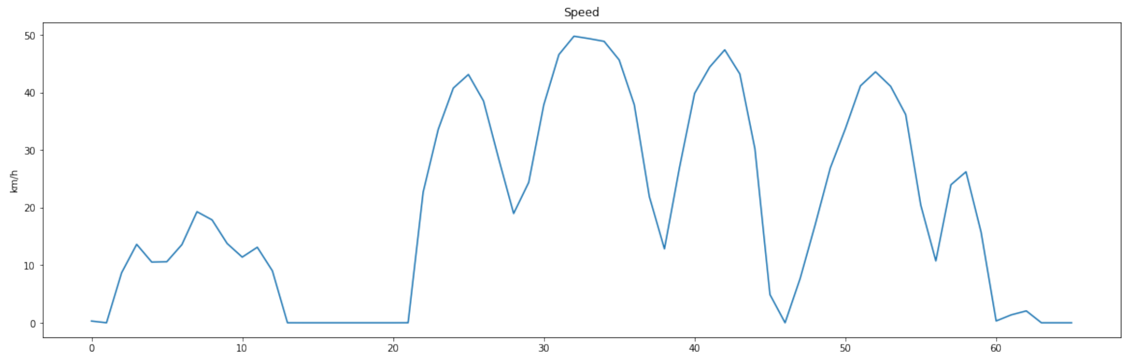
[42]:
```python
some_track_id = track_df['track.id'].unique()[10]
#print(some_track_id)
#print(track_df['track.id'] == some_track_id)
#print("The false track df is:")
some_track = track_df[track_df['track.id'] == some_track_id]
#print("Now the some track is:")
#print(some_track)
some_track.plot(figsize = (10,20))
```

[42]: <matplotlib.axes._subplots.AxesSubplot at 0x23df16b7b88>



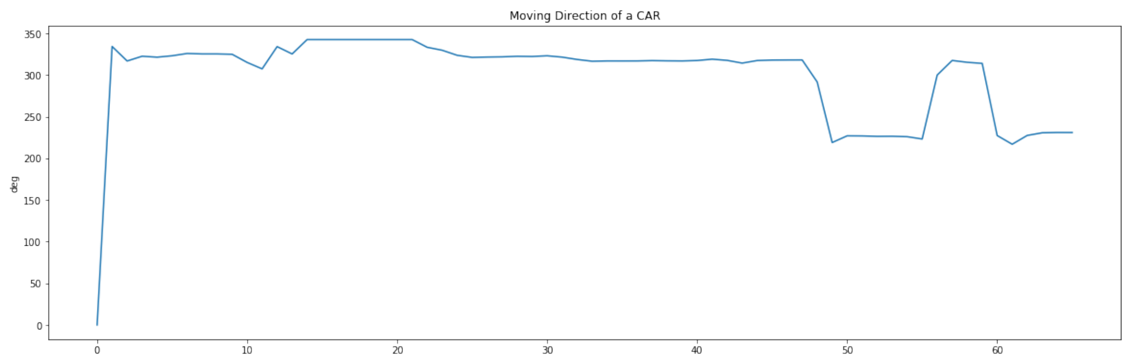
[18]:
```python
ax = some_track['GPS Speed.value'].plot(figsize=(20,6))
ax.set_title("Speed")
ax.set_ylabel(some_track['GPS Speed.unit'][0])
#some_track['GPS Speed.value']
ax
```

[18]: <matplotlib.axes._subplots.AxesSubplot at 0x23dee3d5848>



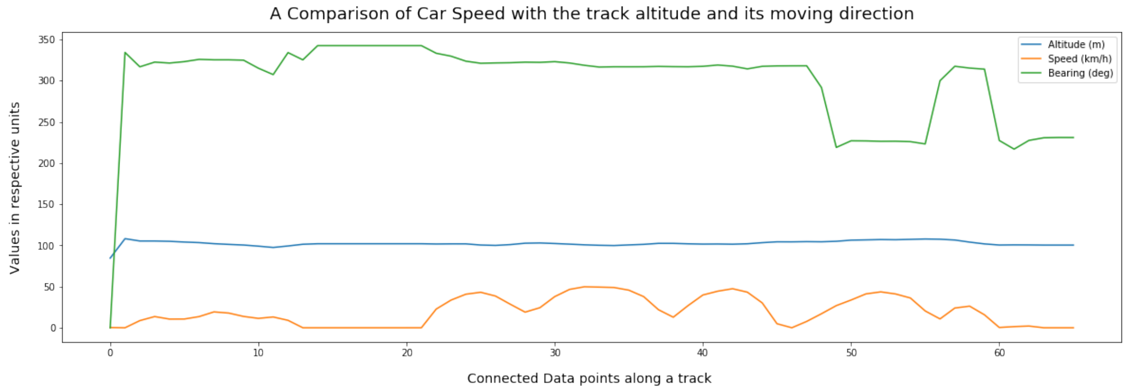
[30]:
```
bx = some_track['GPS Bearing.value'].plot(figsize=(20,6))
bx.set_title("Moving Direction of a CAR")
bx.set_ylabel(some_track['GPS Bearing.unit'][0])
bx
```

[30]: <matplotlib.axes._subplots.AxesSubplot at 0x23df0451308>



[29]:
```
import matplotlib.pyplot as plt

#First prepare the data into dictionary form
df = pd.DataFrame({'Altitude (m)':some_track['GPS Altitude.value'], 'Speed (km/
↪h)':some_track['GPS Speed.value'], 'Bearing (deg)':some_track['GPS Bearing.
↪value']})
#Now plot the data
df.plot(figsize=(20,6))
plt.title("A Comparison of Car Speed with the track altitude and its moving␣
↪direction",y=1.02, fontsize=18)
plt.xlabel("Connected Data points along a track ", labelpad=15, fontsize=14)
plt.ylabel("Values in respective units", labelpad=15, fontsize=14)
```

[29]: Text(0, 0.5, 'Values in respective units')



3.1 Interactive Map

The following map-based visualization makes use of folium. It allows to visualizate geospatial data based on an interactive leaflet map. Since the data in the GeoDataframe is modelled as a set of Point instead of a LineString, we have to manually create a polyline

[60]:
```
import folium

lats = list(some_track['geometry'].apply(lambda coord: coord.y))
lngs = list(some_track['geometry'].apply(lambda coord: coord.x))

avg_lat = sum(lats) / len(lats)
avg_lngs = sum(lngs) / len(lngs)

m = folium.Map(location=[avg_lat, avg_lngs], zoom_start=13)
folium.PolyLine([coords for coords in zip(lats, lngs)], color='black').add_to(m)
m
```

[60]: `<folium.folium.Map at 0x23df330cc08>`

4 Example: Visualization with pydeck (deck.gl)

The pydeck library makes use of the basemap tiles from Mapbox. In case you want to visualize the map with basemap tiles, you need to register with MapBox, and configure a specific access token. The service is free until a certain level of traffic is esceeded.

You can either configure it via your terminal (i.e. `export MAPBOX_API_KEY=<mapbox-key-here>`), which pydeck will automatically read, or you can pass it as a variable to the generation of pydeck (i.e. `pdk.Deck(mapbox_key=<mapbox-key-here>, ...)`.

[66]:
```
import pydeck as pdk

# for pydeck the attributes have to be flat
```

```
track_df['lat'] = track_df['geometry'].apply(lambda coord: coord.y)
track_df['lng'] = track_df['geometry'].apply(lambda coord: coord.x)
vis_df = pd.DataFrame(track_df)
vis_df['speed'] = vis_df['Speed.value']

# omit unit columns
vis_df_cols = [col for col in vis_df.columns if col.lower()[len(col)-4:
↪len(col)] != 'unit']
vis_df = vis_df[vis_df_cols]

layer = pdk.Layer(
    'ScatterplotLayer',
    data=vis_df,
    get_position='[lng, lat]',
    auto_highlight=True,
    get_radius=10,          # Radius is given in meters
    get_fill_color='[speed < 20 ? 0 : (speed - 20)*8.5, speed < 50 ? 255 : 255␣
↪- (speed-50)*8.5, 0, 140]',  # Set an RGBA value for fill
    pickable=True
)

# Set the viewport location
view_state = pdk.ViewState(
    longitude=7.5963592529296875,
    latitude=51.96246168188569,
    zoom=10,
    min_zoom=5,
    max_zoom=15,
    pitch=40.5,
    bearing=-27.36)

r = pdk.Deck(
    width=200,
    layers=[layer],
    initial_view_state=view_state,
    mapbox_key="pk.
↪eyJ1IjoiamFuYWtwYXJhanVsaSIsImEiOiJjaWdtMWd2eWUwMjRvdXJrcjVhbTFvcmszIn0.
↪jRIRtmgCm5waI7RXih3t5A"
)
r.to_html('tracks_muenster.html', iframe_width=900, iframe_height = 500)
```

```
<IPython.lib.display.IFrame at 0x23df3641888>
```

[66]:
```
'D:\\MSC_GeoTech\\Study_Materials\\Course\\Second_Semester\\Floating_Car_Project
\\enviroCar\\envirocar-py\\examples\\tracks_muenster.html'
```

brief description of your experience: what went fine, where did you face problems and how did you overcome the problems?

```
My start of the assignment had to pay a lot of toil in installing the software and its packages
Upon completion of the detailed instructions given in the link above, finally geopandas was suc
Happily, no further problems were faced in course of modification of the given project.
```

screenprint(s) of the last page of your Notebook, presenting the result of your modification..

```
[ ]:
```