大和ハウス工業 スマートロジスティクス
オープンデータチャレンジ

データの利用に向けて

INIAD (東洋大学情報連携学部)

本日の概要

1. API利用手順
2. API利用の基本
3. サンプルプログラム

① API利用までの手順

API利用までの手順

- オープンデータAPIを利用するためには、開発者サイトへのユーザ登録が必要
- 具体的には、以下の手順となります
  1. コンテストサイトにおいて、応募規約・API利用許諾規約に同意
  2. 開発者サイトからユーザ登録
  3. 開発者サイトにおいてアクセストークンの発行（確認）

コンテストサイト

https://daiwa-open-challenge.jp/

- コンテストサイトの下方に「応募規約」および「API利用許諾規約」があります
- こちらの内容をよく読んで頂いた後に、同意のチェックを入れると、ボタンから開発者サイト（＝応募サイト）に入れます

開発者サイト：ユーザ登録

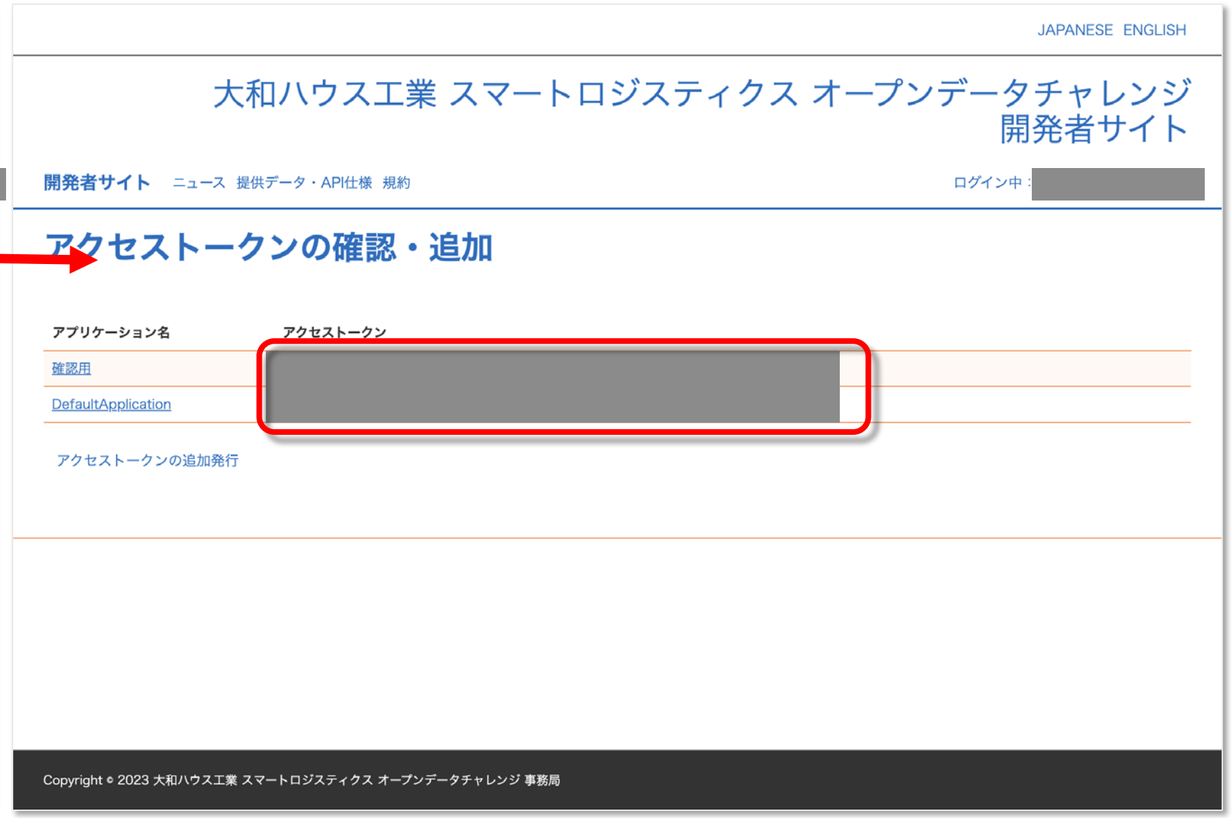
- 開発者サイトでは、まず始めに「ユーザ登録」を行なってください
  - メールアドレスの確認を完了いただいた後、申請内容の確認を行います
  - 登録完了まで最大で2営業日ほどお時間を頂いております

開発者サイト：ログイン

- 事務局での登録内容を確認が完了すると、メールでお知らせします
- ユーザ登録が完了すると、ログインできるようになります

開発者サイト：アクセストークンの確認

- APIアクセスにはアクセストークンが必要となります
- ログイン後、右側の画面にある「アクセストークン」を確認し、手元に控えてください



② API利用の基本

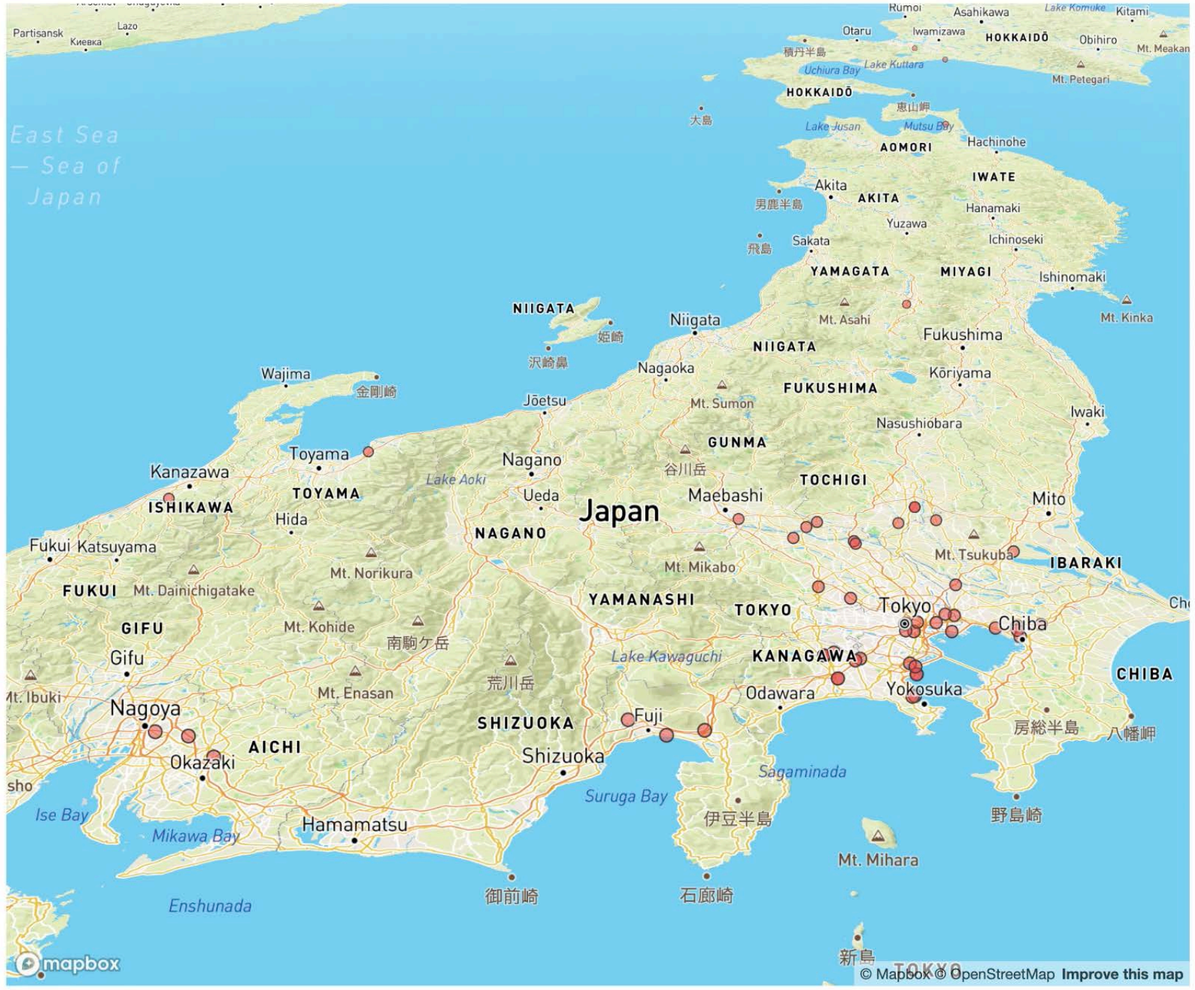
提供するオープンデータ

- 本コンテストでは、以下のデータを、専用API（オープンデータAPI）で提供しています
  - トラックの位置情報
  - トラックの加速度情報
  - ドライバーが出発前後に計測したバイタルデータ
  - ドライバーが運転中に取得したバイタルデータ
  - ヒヤリハットのイベントデータ・ヒヤリハット発生時の映像（※個人が特定できないように加工してあります）
  - 車両マスタ
  - ユーザマスタ
- 警察庁より
  - 交通事故統計情報（2019-2021）

トラックの位置情報の例

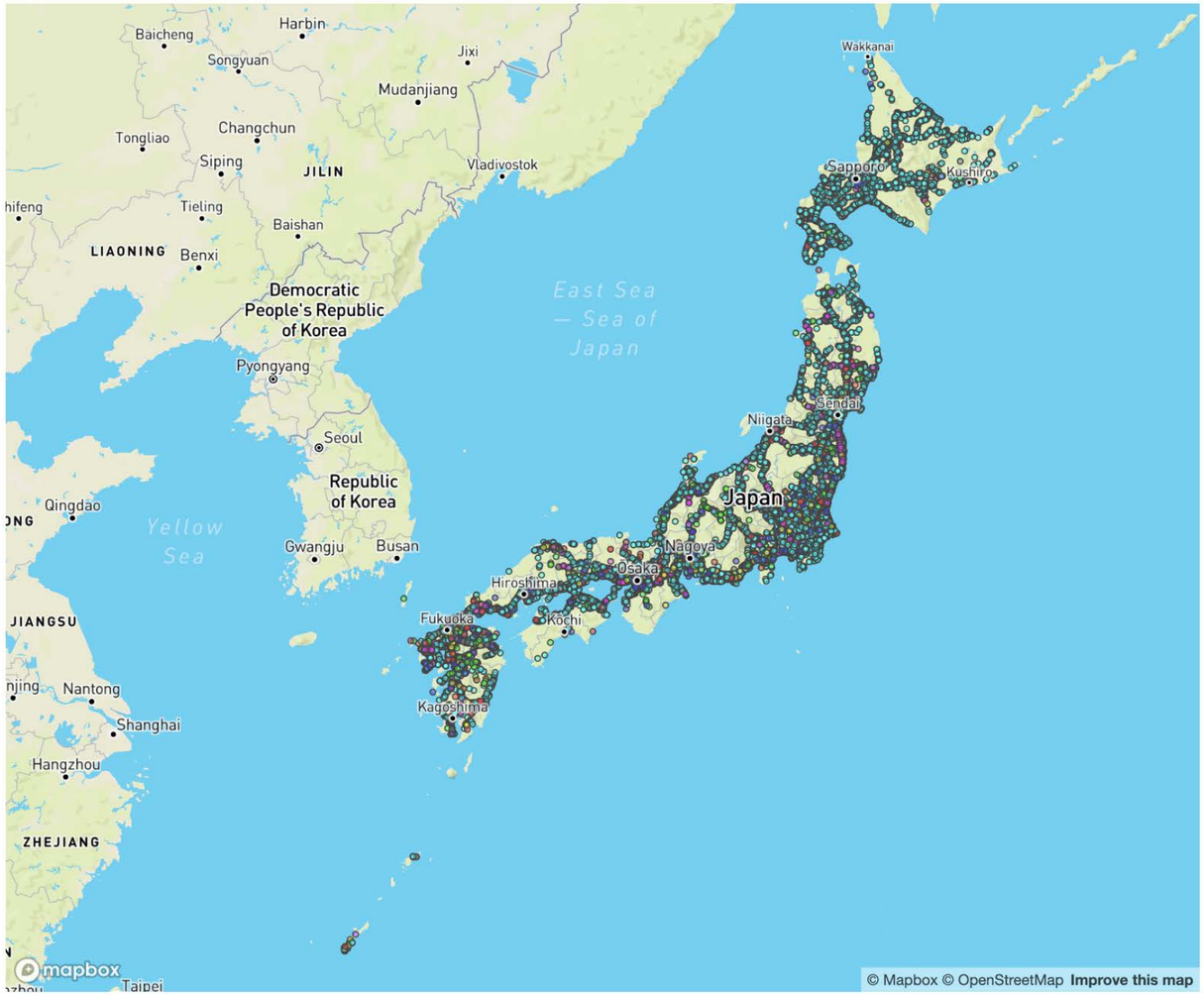
2022年10月31日の実際のトラックの移動履歴を表示

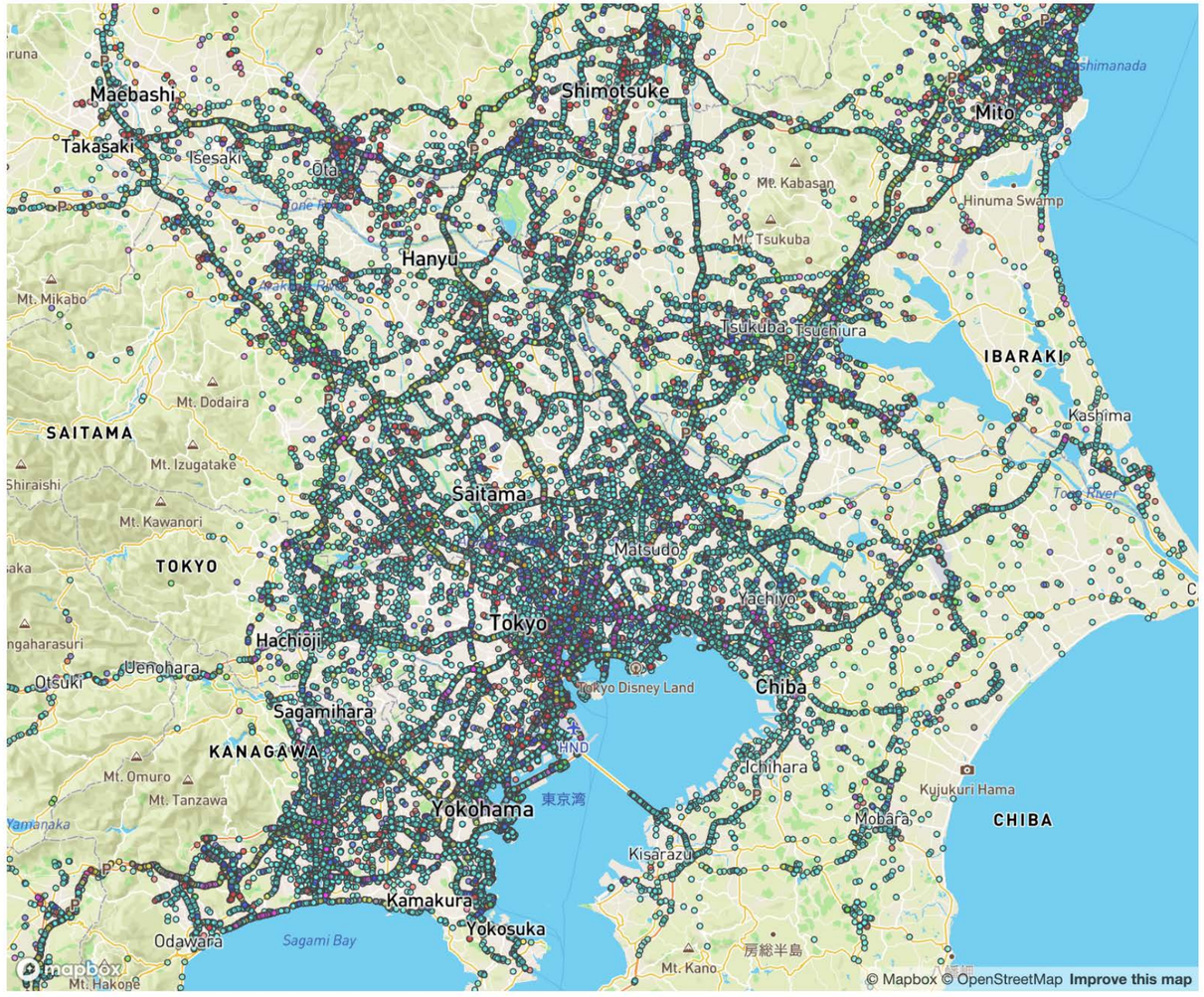
2022/10/31 0:02



ヒヤリハットのイベントデータ

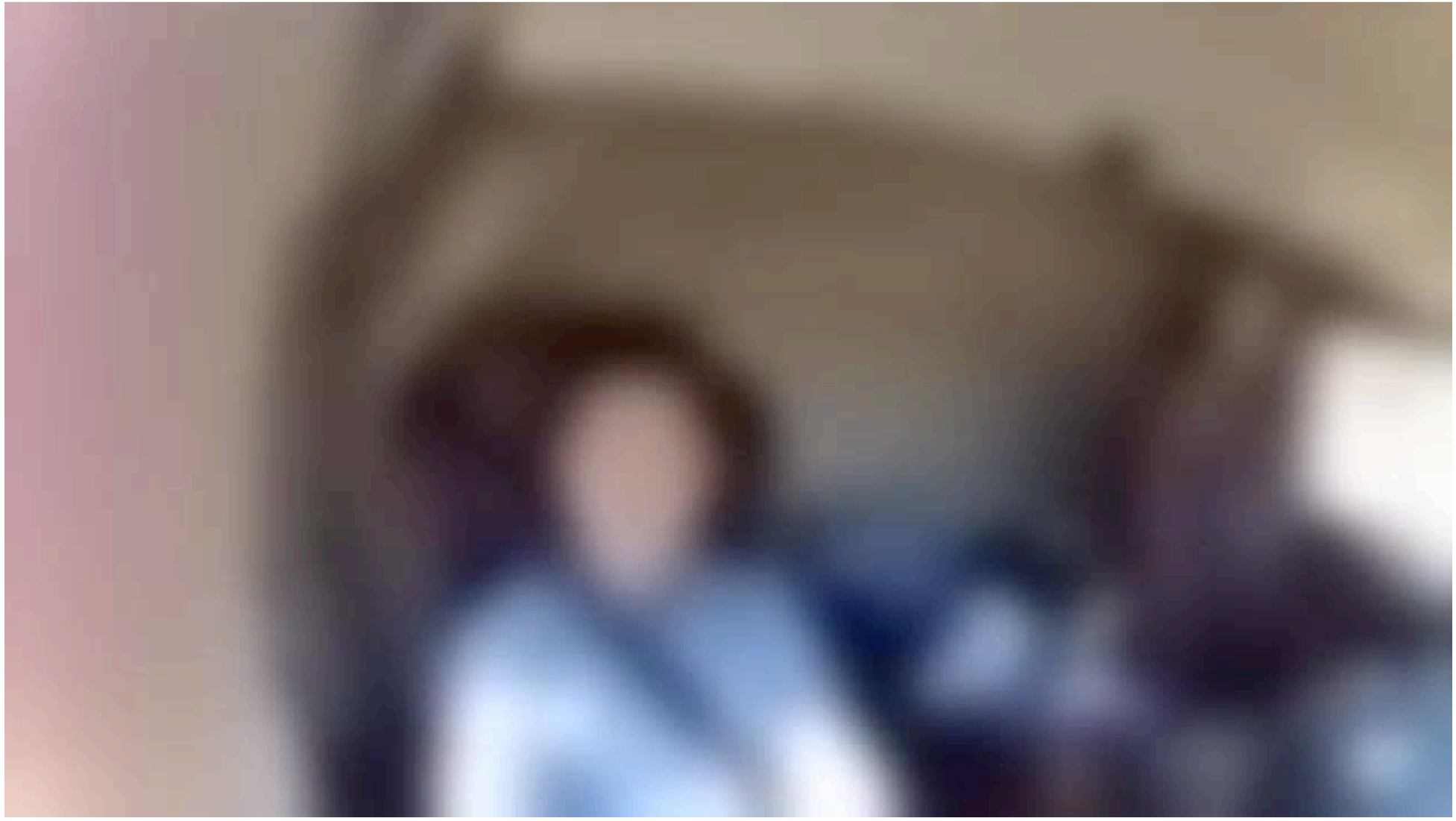
2022年1月～10日の合計16万件を表示





ヒヤリハット発生時の映像（車外）

ヒヤリハット発生時の映像（車内）



オープンデータAPIの利用方法

- 本コンテストでは、次のような仕様のAPIを提供します
  - URL：https://api.daiwa-open-challenge.jp/files/ から始まるURLを対象データごとに指定します
  - メソッド：HTTPのGETメソッドでアクセスします
  - パラメータ：URLパラメータ acl:consumerKey として、アクセストークンを指定する必要があります
- レスポンスは、ZIP圧縮されたCSVファイルです
  - 個々のデータがかなりのサイズとなるため、このようなAPI形式となっています
  - ※どのような形でデータを切り出すかといった点も、開発者の皆様にぜひ検討をしていただければと考えています

取得したいデータのURLを確認する

- 開発者サイトで、提供データごとにURLのパスが掲載されています
- API仕様のページに、詳細情報が掲載されています

JAPANESE ENGLISH

大和ハウス工業 スマートロジスティクス オープンデータチャレンジ
開発者サイト

開発者サイト ニュース 提供データ・API仕様 規約 ログイン中： Account

提供データ・API仕様

提供するオープンデータ

大和ハウス工業 スマートロジスティクス オープンデータチャレンジでは、以下のデータを提供します。

データの種類	提供する期間
トラックの位置情報	2022/01/01 ～ 2022/12/31
トラックの加速度情報	2022/09/01 ～ 2022/12/31
ドライバーが出発前後に計測したバイタルデータ	2022/01/01 ～ 2023/04/30
ドライバーが運転中に取得したバイタルデータ	2022/04/21 ～ 2023/04/30 ただし、以下の日のデータはありません。 • 2022/04/23, 2022/04/24 • 2022/05/01, 2022/05/03, 2022/05/04, 2022/05/05, 2022/05/07, 2022/05/08, 2022/05/22 • 2022/06/05, 2022/06/12, 2022/06/19, 2022/06/25, 2022/06/26 • 2022/07/03, 2022/07/31 • 2022/08/28 • 2022/09/04, 2022/09/18, 2022/09/19 • 2022/10/01, 2022/10/02, 2022/10/09, 2022/10/15, 2022/10/23, 2022/10/29, 2022/1

Copyright © 2023 大和ハウス工業 スマートロジスティクス オープンデータチャレンジ 事務局

トラックの位置情報

定期的に取得したトラックの位置情報を、CSVファイル（文字コードはutf-8）をZIPで圧縮した形式で取得できます。

データ取得先URLは

```
https://api.daiwa-open-challenge.jp/files/sensors_map/sensors_map_{yyyymmdd}.zip?acl:consumerKey=[アクセストークン]
```

です。ただし、{yyyymmdd}は、取得対象の日付です。

※ 1つのZIPファイルのサイズは、0.1G～1.5Gbytes程度です。解凍して得られるCSVファイルのサイズは、0.5G～4.5GBytes程度です。日毎のファイルサイズについてはこちら。

CSV内各カラムは、以下の意味を持ちます。

カラム名	意味
date	発生した日付
company_id	企業識別子
office_id	拠点識別子
car_number	車両識別子
user_id	ユーザ識別子（一部マスキングされています）
timestamp	発生した日時（マイクロ秒単位・日本標準時）
latitude	緯度（世界測地系）
longitude	経度（世界測地系）
speed	スピード
mapped_latitude	マップマッチ後の緯度（世界測地系）
mapped_longitude	マップマッチ後の経度（世界測地系）
distance	走行距離[m]（ドライブレコーダーが認識する、区間連続走行距離）
d_kbn	ドライブレコーダーの種類であり、1または2の値をとります。

Copyright © 2023 大和ハウス工業 スマートロジスティクス オープンデータチャレンジ 事務局

（例）トラックの位置情報を取得する

https://developer.daiwa-open-challenge.jp/documents#sensors_map

- トラックの位置情報の場合は、以下のパスとなります
  - https://api.daiwa-open-challenge.jp/files/sensors_map/sensors_map_{yyyymmdd}.zip?acl:consumerKey=[アクセストークン]
  - {yyyymmdd}の箇所には、取得したいデータ日時を指定します

トラックの位置情報

定期的に取得したトラックの位置情報を、CSVファイル（文字コードはutf-8）をZIPで圧縮した形式で取得できます。

データ取得先URLは、

https://api.daiwa-open-challenge.jp/files/sensors_map/sensors_map_{yyyymmdd}.zip?acl:consumerKey=[アクセストークン]

です。ただし、{yyyymmdd}は、取得対象の日付です。
※ 1つのZIPファイルのサイズは、0.1G～1.5Gbytes程度です。解凍して得られるCSVファイルのサイズは、0.5G～4.5GBytes程度です。日毎のファイルサイズについてはこちら。

CSV内各カラムは、以下の意味を持ちます。

カラム名	意味
date	発生した日付
company_id	企業識別子
office_id	拠点識別子
car_number	車両識別子
user_id	ユーザ識別子（一部マスキングされています）
timestamp	発生した日時（マイクロ秒単位・日本標準時）
latitude	緯度（世界測地系）
longitude	経度（世界測地系）
speed	スピード
mapped_latitude	マップマッチ後の緯度（世界測地系）
mapped_longitude	マップマッチ後の経度（世界測地系）
distance	走行距離[m]（ドライブレコーダーが認識する、区間連続走行距離）
d_kbn	ドライブレコーダーの種類であり、1または2の値をとります。

Copyright © 2023 大和ハウス工業 スマートロジスティクス オープンデータチャレンジ 事務局

（例）トラックの位置情報を取得する

https://developer.daiwa-open-challenge.jp/documents#sensors_map

- 例えば、2022年4月1日のトラックの位置情報を取得するためには、以下のようなURLを組立て、HTTP GETで取得します
  - https://api.daiwa-open-challenge.jp/files/sensors_map/sensors_map_20220401.zip?acl:consumerKey=[アクセストークン]

✓Webブラウザで上記URLを開くとデータがダウンロードできます

✓curlであれば、以下のようなコマンドでダウンロードできます

- curl -L -o sensors_map_20220401.zip https://api.daiwa-open-challenge.jp/files/sensors_map/sensors_map_20220401.zip?acl:consumerKey=[アクセストークン]

③ サンプルプログラム

本日のサンプルコードは、
以下からダウンロード可能です

bit.ly/dcs230523

サンプルの前提

- ここでは、Google Colaboratory を利用して、Pythonでオープンデータをダウンロードして処理するサンプルプログラムを、いくつかお示ししします

① トラックの移動履歴を、OpenStreetMap上で表示する

② ヒヤリハットのイベントデータを、 OpenStreetMap上で表示する

③ 物体検出をドライブレコーダの動画に適用する

④ ドライブレコーダの動画を高解像度化する

③-1 トラックの位置情報の可視化

visualize_tracks.ipynb

ライブラリのインポート

1. 準備

1-a) ライブラリのインポート

ここでは、以下のライブラリをインポートします。

- requests : HTTPを用いてURLからコンテンツをダウンロードするライブラリ
- BytesIO : ストリームを扱う標準ライブラリ
- ZipFile : ZIPアーカイブを展開する標準ライブラリ
- Pandas : 表形式のデータを扱うためのライブラリ
- GeoPandas : 地理上を扱うためのPandasの拡張
- shapely.geometry : 地理情報を扱うためのライブラリ

```
[ ]  !pip install geopandas
     !pip install folium mapclassify
```

```
[ ]  import requests
     from zipfile import ZipFile
     from io import BytesIO
     import pandas as pd
     import geopandas as gpd
     from shapely.geometry import Point, LineString, shape
```

定数の定義

- アクセストークンには、ご自身のものを設定してください

1-b) 定数の定義

ここでは、以下を定義します。

- ACCESS_TOKEN : 開発者サイトで発行したアクセストークン
- API_URL_SENSORS : トラックの位置情報を取得するためのURL

```
[ ]  ACCESS_TOKEN = 'YOUR ACCESS TOKEN'
     API_URL_SENSORS = 'https://api.daiwa-open-challenge.jp/files/sensors_map/sensors_map_{}.zip?acl:consumerKey={}'
```

データのダウンロード・展開

2. データの展開

2-a) データのダウンロード

ここでは、実際に4月1日のデータをダウンロードします。

```
[ ]  url = API_URL_SENSORS.format('20220401', ACCESS_TOKEN)

     r = requests.get(url)
```

2-b) ZIPの展開

ダウンロードしたデータはZIP形式で圧縮されているので、ZipFileとして読み込みます。

```
[ ]  zip = ZipFile(BytesIO(r.content), 'r')
```

2-c) 内容の確認

内容を確認します。

```
[ ]  infolist = zip.infolist()
     infolist
```

Pandasでの読み込み

2-d) Pandasでの読み込み

このファイルはCSV形式なので、Zipを展開し、Pandasで読み込みます。

```
[ ]  with zip.open(infolist[0]) as csv_file:
       df = pd.read_csv(csv_file)

     zip.close()

     df
```

Pandasでの読み込み

- しばらくすると、以下のように表形式で読み込みが完了します

	date	company_id	office_id	car_number	user_id	timestamp	latitude	longitude	speed
0	20220401	VTC0001	VTC000100003	JNCMN00A0HU023228	_l2tyms1imcnA8tRZ	2022/04/01 00:00:04.000000	33.772527	130.997174	0.018520
1	20220401	ZAX0200	ZAX020000005	NPR85-7061850	_7VinmBQwstsAG	2022/04/01 00:00:20.000000	35.778706	139.720173	0.690384
2	20220401	ZAX2401	ZAX240100013	FRR90-7006010	_6dCDh8eQOgMXi	2022/04/01 00:00:38.000000	36.464611	136.597555	0.076652
3	20220401	ZAX0100	ZAX010019453	JNCMM90G4GU012804	_cOjXvcPhr1snw	2022/04/01 00:01:00.000000	39.277840	141.078453	21.287712
4	20220401	VCL0001	VCL000100002	FS74HZ-511853	_ALMB1u41pbpahkEu	2022/04/01 00:01:02.000000	34.738554	135.613520	14.682245
...	...	...	...	...	...	...	...	...	...
21384962	20220401	VCL0001	VCL000100007	JNCMM60G5GU010106	_IGSdp5RgHN1HWaiQ	2022/04/01 22:22:49.000000	35.331831	139.368097	15.371599
21384963	20220401	ZAX2401	ZAX240100007	FRR90-7094695	_kHnpmaGo8flSt	2022/04/01 22:22:50.000000	35.127646	138.800723	1.692008
21384964	20220401	ZAX2401	ZAX240100022	FRR90-7045060	unknown	2022/04/01 22:22:50.000000	35.042707	138.485628	14.863843
21384965	20220401	VCL0001	VCL000100007	JNCMM60G5GU010106	_IGSdp5RgHN1HWaiQ	2022/04/01 22:22:52.000000	35.331801	139.367581	15.541882
21384966	20220401	VCL0001	VCL000100006	CG5ZA-00306	unknown	2022/04/01 22:22:53.000000	35.537757	139.422052	13.091068

21384967 rows × 13 columns

トラックの台数の確認

- car_number のユニーク数を数えると、1623台のトラックが走行していることが確認できます

2-e) トラックの台数の確認

ユニークなトラックの台数を確認してみます。

```
[8]  len(df['car_number'].unique())
```

```
1623
```

データの下処理

3. データの下処理

3-a) タイムスタンプの変換

タイムスタンプを datetime 型に変換し、時・分・秒のカラムを作成します。

```
[ ]  df['timestamp'] = pd.to_datetime(df['timestamp'])
     df['hour'] = df['timestamp'].dt.hour
     df['minute'] = df['timestamp'].dt.minute
     df['second'] = df['timestamp'].dt.second
     df
```

3-b) ドライブでの分割

トラックの位置情報が、10分以上途切れている場合は、別のIDを割り当てます。

```
[ ]  df = df.sort_values(['car_number', 'timestamp'])
     df['change_car'] = df['car_number'].ne(df['car_number'].shift())
     df['change_ts'] = df['timestamp'].diff().dt.total_seconds() > 600
     df['drive_id'] = (df['change_car'] | df['change_ts']).cumsum()
     df = df.drop('change_car', axis=1)
     df = df.drop('change_ts', axis=1)
     df
```

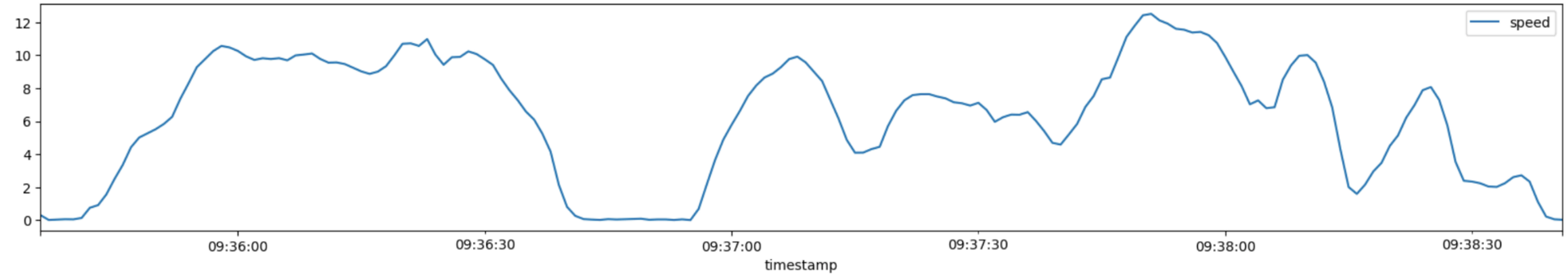
トラックの運行（速度・走行距離）の可視化

3-c) トラックの速度の可視化。

ここでは、ID=4のドライブのトラックの速度の推移を可視化します。

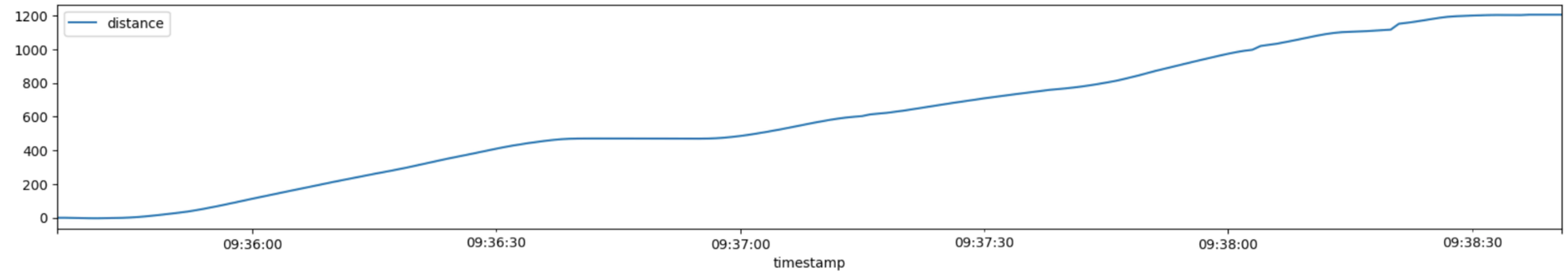
```
[11] df[df['drive_id'] == 4].plot('timestamp', 'speed', figsize=(20,3))
```

<Axes: xlabel='timestamp'>




```
[12] df[df['drive_id'] == 4].plot('timestamp', 'distance', figsize=(20,3))
```

<Axes: xlabel='timestamp'>



ジオメトリカラムの追加

4. 地図上での可視化

4-a) 分単位での集約

地図上の可視化のために、時・分で集約し、それぞれ1行目を取り出します。

```
[ ]  df_m = df.groupby(['car_number', 'hour', 'minute'], sort=False).first()
     df_m = df_m.reset_index()
     df_m
```

4-b) ジオメトリカラムの追加

mapped_latitude および mapped_longitude カラムをもとに、緯度軽度のカラムを作成します。

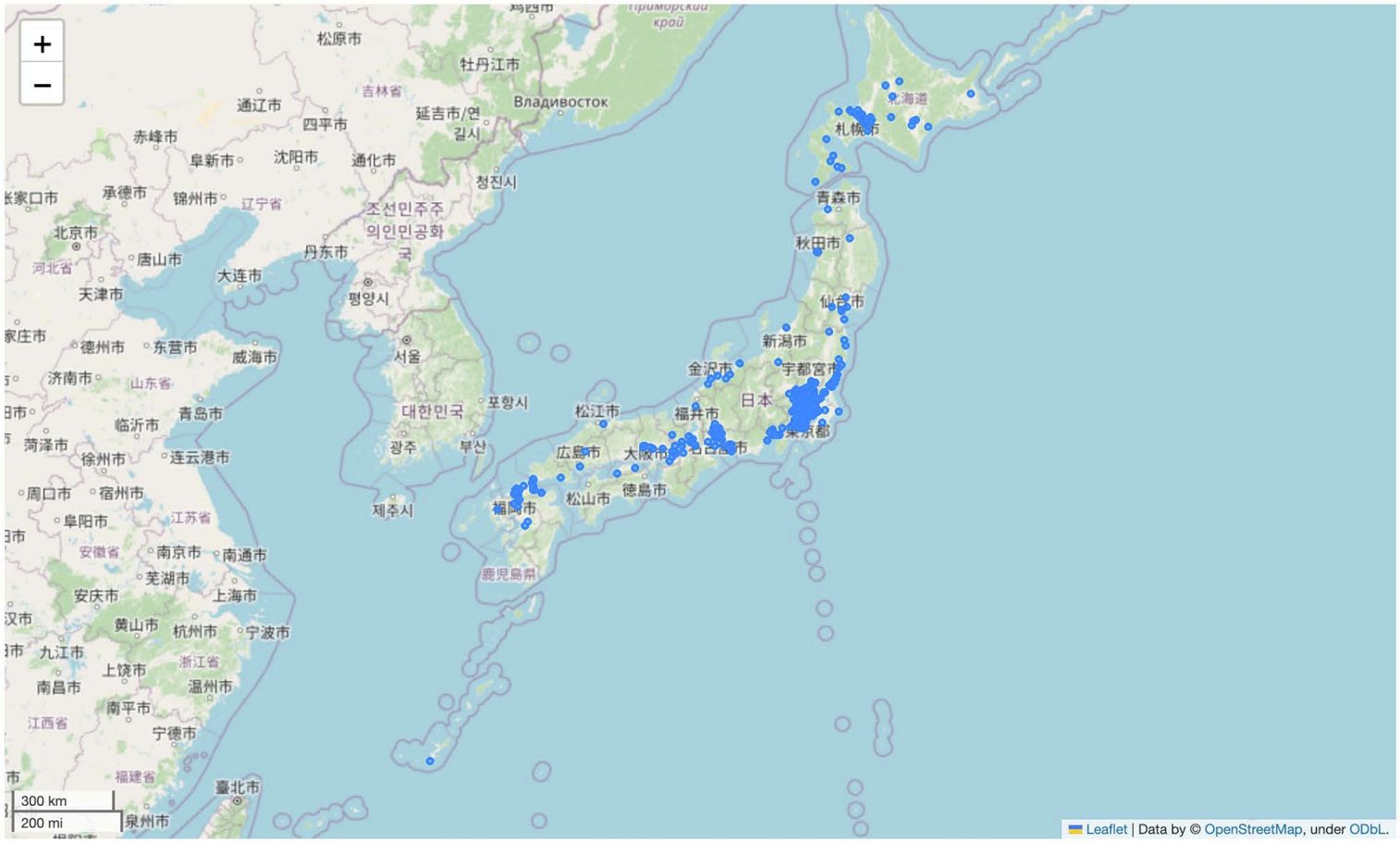
```
[ ]  gdf = gpd.GeoDataFrame(df_m, geometry=gpd.points_from_xy(df_m['mapped_longitude'], df_m['mapped_latitude']), crs=4326)
     gdf
```

正午の状況の表示

4-c) 正午の状況の表示

正午のトラックの位置を表示してみます。

```
[16] gdf_noon = gdf[(gdf['hour'] == 12) & (gdf['minute'] == 0)][['car_number', 'geometry']]
     gdf_noon.explore(tiles='OpenStreetMap')
```



トラックの位置情報の時系列での集約

4-d) トラックの位置情報の時系列での集約

トラックの位置情報を車両ごとに集約し、トラックの軌跡をもつ DataFrame を作成します。

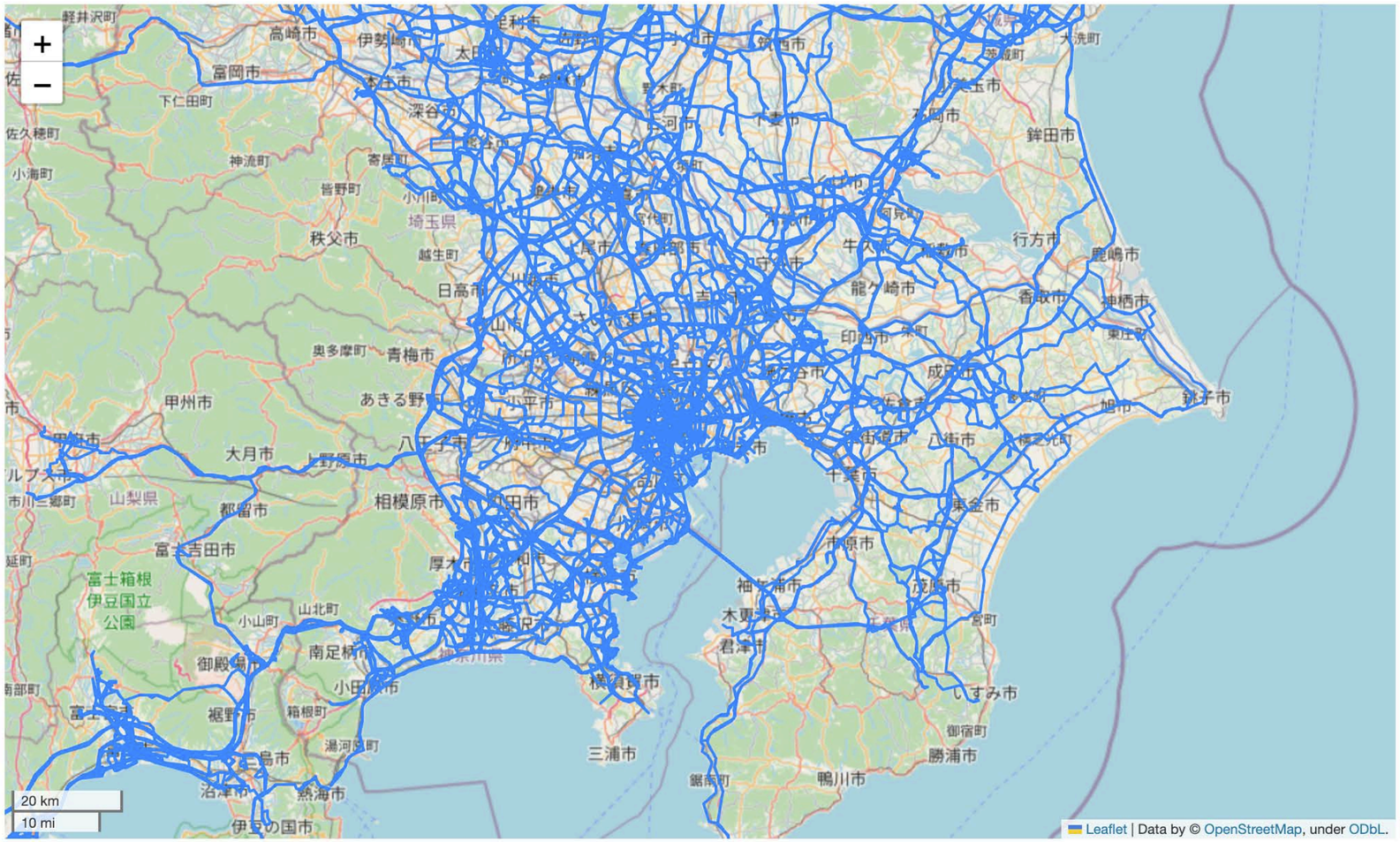
```
[18] def to_line(pts):
         if len(pts) <= 1:
             return None
         ls = LineString(pts.tolist())
         return ls

     lines = gdf.groupby('car_number')['geometry'].apply(to_line)
     gdf_grouped = gpd.GeoDataFrame(lines, crs='4326')
     gdf_grouped = gdf_grouped.dropna()
     gdf_grouped
```

トラックの軌跡の表示

4-e) トラックの軌跡の表示

トラックの軌跡を表示してみます。

```
gdf_grouped.explore(tiles='OpenStreetMap')
```



③-2 ヒヤリハットの可視化

visualize_events.ipynb

ライブラリのインポート

1. 準備

1-a) ライブラリのインポート

ここでは、以下のライブラリをインポートします。

- requests : HTTPを用いてURLからコンテンツをダウンロードするライブラリ
- BytesIO : ストリームを扱う標準ライブラリ
- ZipFile : ZIPアーカイブを展開する標準ライブラリ
- Pandas : 表形式のデータを扱うためのライブラリ
- GeoPandas : 地理上を扱うためのPandasの拡張
- Google Drive に接続するためのライブラリ

```
[ ]  !pip install geopandas
     !pip install folium mapclassify
```

```
[ ]  import requests
     from zipfile import ZipFile
     from io import BytesIO
     import pandas as pd
     import geopandas as gpd
     from google.colab import drive
```

定数の定義

- アクセストークンには、ご自身のものを設定してください

1-b) 定数の定義

ここでは、以下を定義します。

- ACCESS_TOKEN : 開発者サイトで発行したアクセストークン
- API_URL_EVENTS : ヒヤリハット情報を取得するためのURL
- DATE : 対象日

```
[ ]  ACCESS_TOKEN = 'YOUR ACCESS TOKEN'
     API_URL_EVENTS = 'https://api.daiwa-open-challenge.jp/files/events/events_{}.zip?acl:consumerKey={}'
     DATE = '20220401'
```

データのダウンロード・Pandasでの読み込み

2. データの展開

2-a) データのダウンロード

ここでは、実際に4月1日のデータをダウンロードします。

```
[ ]  url = API_URL_EVENTS.format(DATE, ACCESS_TOKEN)

     r = requests.get(url)
```

2-b) ZIPの展開

ダウンロードしたデータはZIP形式で圧縮されているので、ZipFileとして読み込みます。

```
[ ]  zip = ZipFile(BytesIO(r.content), 'r')
```

2-c) Pandasでの読み込み

このファイルはCSV形式なので、Zipを展開し、Pandasで読み込みます。

```
[ ]  with zip.open('events_{}.csv'.format(DATE)) as csv_file:
       df = pd.read_csv(csv_file)

     df
```

Pandasでの読み込み

- しばらくすると、以下のように表形式で読み込みが完了します

	date	company_id	office_id	car_number	user_id	timestamp	event_id	event_r
0	20220401	VEL0001	VEL000100003	FW1AH-108630	_pvvSdFR6v6pmin8F	2022/04/01 00:06:50.000	800106	一時イ
1	20220401	ZAX2901	ZAX290100085	GP2-3132370	_8iwh3aWHs9lmmnux7	2022/04/01 00:07:40.000	800102	急
2	20220401	VCL0001	VCL000100006	JNCMBP0GXJU032864	_6A7lKhkqNy6cu	2022/04/01 01:23:23.000	800105	
3	20220401	VEL0001	VEL000100003	CYJ77W8-7002425	_L5SiFuZJMCXEbO3u	2022/04/01 01:32:32.000	800106	一時イ
4	20220401	ZAX0100	ZAX0100A0001	HNT32-181386	unknown	2022/04/01 02:47:34.000	800105	
...	...	...	...	...	...	...	...	
855	20220401	ZAX2901	ZAX290100064	ZVW30-5650548	unknown	2022/04/01 21:54:59.000	800102	急
856	20220401	ZAX2401	ZAX240100022	FC2AB-127976	_LVplbUZTuMC8O	2022/04/01 23:01:55.000	800102	急
857	20220401	ZAX7201	ZAX720100011	FY64VY-520059	_JskAWtp77uMY2AOl	2022/04/01 23:11:23.000	800105	
858	20220401	ZAX2401	ZAX240100117	NSP160-0013632	unknown	2022/04/01 23:25:55.000	800106	一時イ
859	20220401	ZAX2901	ZAX290100002	FTR90-7012141	_2AyZgSU7xouWCQ5	2022/04/01 23:40:31.000	800105	

860 rows × 13 columns

ジオメトリカラムの追加

3. 地図上で可視化

3-a) ジオメトリカラムの追加

latitude および longitude カラムをもとに、緯度経度のカラムを作成します。

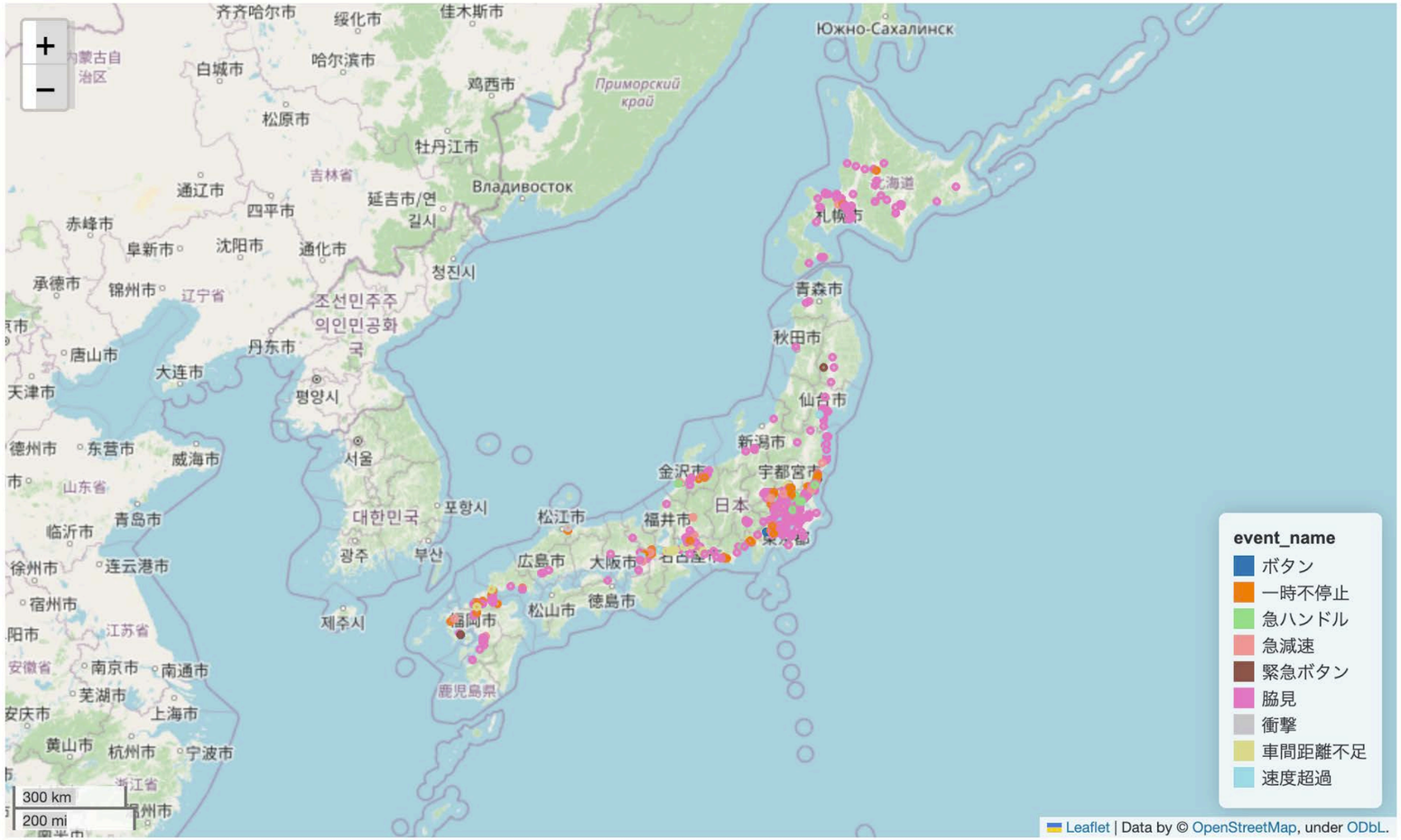
```
[ ]  gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df['longitude'], df['latitude']), crs=4326)
     gdf
```

地図上への表示

3-b) 地図上に表示

ヒヤリハットの位置を、event_name で色分けして表示してみます。

```
[10] gdf.explore(tiles='OpenStreetMap', column='event_name')
```



Google Drive への動画の展開

- 以下のコードを実行すると、Google Driveのdaiwaフォルダ以下にZIPファイルが展開されます

4. Google Drive に動画データを展開

4-a) Google Drive のマウント

Google Driveをマウントします。

```
[ ]  drive.mount('/content/drive')
```

4-b) ZIPファイルの展開

Google Drive の daiwa/20220401 フォルダに、ZIPファイルを展開します。

```
[ ]  zip.extractall('/content/drive/MyDrive/daiwa/{}'.format(DATE))
```

4-c) ZIPファイルのクローズ

ZIPファイルをクローズします。

```
[ ]  zip.close()
```

③-3 ドライブレコーダの動画に物体検出を適用する

object_detection.ipynb

作業の概要

- ヒヤリハットの動画データは、個人が特定できないような加工がなされています
- この動画に、YOLOv8の物体検出を適用してみましょう
- YOLOv8とは…
  - Ultralytics社が公開している、物体検出のライブラリ
  - 学習済みのモデルが公開されている
  - https://github.com/ultralytics/ultralytics
- ここでは、学習済みのYOLOv8nモデルを、そのまま適用してみます

オリジナル動画

検出結果

③-4 ドライブレコーダの動画の高解像度化を試す

super_resolution.ipynb

作業の概要

- ヒヤリハットの動画データは、個人が特定できないような加工がなされています
- この動画の高解像度化を試してみましょう
- AIを活用した高解像度化のライブラリとして、さまざまなモデルが公開されています
- ここでは、SwinIRというライブラリを用いて高解像度化を試してみました
  - https://github.com/JingyunLiang/SwinIR

オリジナル動画

高解像度化の結果

本日のサンプルコードは、
以下からダウンロード可能です

bit.ly/dcs230523