All packages (/)

Module `usgs_lidar`

Classes

```
class UsgsLidar (path='https://s3-us-west-2.amazonaws.com/usgs-lidar-public/',
                 pipeline_json_path: str = '../pipeline.json')
```

A class that load, fetch, visualise, and transform publicly available LIDAR data on AWS.

Args

`path : str`, optional
: [url path location of the Lidar data]. Defaults to "https://s3-us-west-2.amazonaws.com/usgs-lidar-public/"

`pipeline_json_path : str`, optional
: [the json file with the pipeline structure]. Defaults to "../pipeline.json".

Returns

`[None]`
: [nonetype object].

Methods

```
def convert_epsg(self, df: geopandas.geodataframe.GeoDataFrame, column: str,
                 epsg_inp=4326, epsg_out=3857)
                 -> geopandas.geodataframe.GeoDataFrame
```

A method that converts EPSG coordinate system

Args

`df : gpd.GeoDataFrame`
: [a geopandas dataframe containing columns of elevation and geometry.]

`column : str`
: [the column geometry.]

`epsg_inp : int`
: [the current geometry EPSG type.]

`epsg_out : int`

[EPSG type the geometry will be converted to.]

Returns

`[Geopandas.GeoDataFrame]`
[a geopandas dataframe]

```
def create_gpd_df(self, epsg, pipe) -> geopandas.geodataframe.GeoDataFrame
```

A method to create geopandas dataframe from a pipeline object

Args

`epsg : int` , optional
[EPSG coordinate system].

`pipe : pdal.Pipeline`
[pipeline object].

Returns

`[Geopandas.GeoDataFrame]`
[a geopandas dataframe].

```
def execute_pipeline(self, polygon: shapely.geometry.polygon.Polygon,
                     epsg=4326, region: str = 'IA_FullState') -> None
```

A method to execute a pipeline and fetch data.

Args

`polygon : Polygon`
[A polygon object].

`epsg : int` , optional
[EPSG coordinate system] Default to 4326.

`region : str` , optional
[the filename of the region] Default to IA_FullState.

Returns

`[None]`
[nonetype object].

```
def fetch_data(self, polygon: shapely.geometry.polygon.Polygon,
               region='IA_FullState') -> dict
```

A method to fetch the data of a region.

Args

`polygon : Polygon`
: [a polygon object].

`region : str`, optional
: [the region where the data will be extracted from].

Returns

`[dict]`
: [a dictionary object with year, geopandas dataframe pair].

```
def fetch_metadata(self) -> pandas.core.frame.DataFrame
```

A method to create metadata for EPT files available on AWS.

Returns

`[pandas.DataFrame]`
: [dataframe of the metadata].

```
def fetch_name_and_year(self, location: str) -> tuple
```

A method to fetch name and year from file name.

Args

`location : str`
: [location of file].

Returns

`[tuple]`
: [tuple of name and year]

```
def fetch_pipeline(self, region: str,
                   polygon: shapely.geometry.polygon.Polygon)
                   -> pdal.pipeline.Pipeline
```

A method to fill the empty values in the json pipeline and create pdal pipeline object

Args

`region : str`
: [the filename of the region].

`polygon`
: (Polygon): [the input polygon].

Returns

`[pdal.pipeline]`
: [pdal pipeline object].

```
def fetch_polygon_boundaries(self,
                             polygon: shapely.geometry.polygon.Polygon)
                             -> tuple
```

A method that fetch the polygon boundaries based on the input polygon

Args

`polygon : Polygon`
: [the input polygon]

Returns

`[tuple]`
: [bounds and polygon exterior coordinates string]

```
def fetch_region_data(self, polygon: shapely.geometry.polygon.Polygon,
                      epsg=4326) -> geopandas.geodataframe.GeoDataFrame
```

A method to fetch the data of a region.

Args

`polygon : polygon`
: [a polygon object].

`epsg : int` , optional
: [EPSG coordinate system].

Returns

`[Geopandas.GeoDataFrame]`

[a geopandas dataframe].

```
def fetch_regions(self, polygon: shapely.geometry.polygon.Polygon,
                  epsg=4326) -> list
```

A method to fetch region(s) within a polygon.

Args

`polygon : Polygon`
[a polygon object].

`epsg : int` , optional
[EPSG coordinate system].

Returns

`[list]`
[lists of regions within the polygon].

```
def load_heatmap(self, png_path: str) -> None
```

A method to load a saved image.

Arg

png_path (str): [the path of the image to load].

Returns

`[None]`
[nonetype object]

```
def plot_terrain(self, gdf: geopandas.geodataframe.GeoDataFrame,
                 fig_size: tuple = (12, 10), size: float = 0.01) -> None
```

A method to plot points in geopandas dataframe as a 3D scatter plot.

Args

`gdf : GeoDataFrame`
[a geopandas dataframe containing columns of elevation and geometry].

`fig_size : tuple` , optional
[filesze of the figure to be displayed]. Defaults to (12, 10)].

`size : float`, optional
: [size of the points to be plotted]. Defaults to 0.01].

Returns

`[None]`
: [nonetype object].

```
def read_csv(self, csv_path, missing_values=['n/a', 'na', 'undefined'])
            -> pandas.core.frame.DataFrame
```

A method to read a csv file

Args

`csv_path : string`
: [the location of the csv file.]

missing_values(string, optional): [null expressions.]

Returns

`[pandas.DataFrame]`
: [pandas dataframe]

```
def read_json(self, json_path: str)
```

A method to read a json file

Args

`json_path : str`
: [the location of the json file].

```
def read_txt(self, txt_path: str) -> list
```

A method to read text file.

Args

`txt_path : str`
: [path to the text file].

Returns

`[list]`
: [list of text files.]

```
def save_heatmap(self, df: geopandas.geodataframe.GeoDataFrame,
                 png_path: str, title: str) -> None
```

A method to plot and save a heatmap.

Args

`df : GeoDataFrame`
: [a geopandas dataframe containing columns of elevation and geometry].

`png_path : str`
: [the path to save the heatmap as PNG].

`title : str`
: [the tite of the image].

Returns

`[None]`
: [nonetype object].

```
def subsample(self, gdf: geopandas.geodataframe.GeoDataFrame, res: int = 3)
              -> geopandas.geodataframe.GeoDataFrame
```

A method to sample a point cloud data by implementing a decimation and voxel grid sampling to reduce point cloud data density.

Args

`gdf : gpd.GeoDataFrame`
: [a geopandas dataframe containing columns of elevation and geometry.]

`res : int` , optional
: [resolution]. Defaults to 3.

Returns

`[Geopandas.GeoDataFrame]`
: [a geopandas dataframe]

Index

Classes