imputation_01

July 2, 2025

[ ]:
```
# CONFIG CELL
from notebook_utils import set_root_directory

set_root_directory()
```

[ ]:
```
import geopandas as gpd
import pandas as pd

from app import constants
from app.imputers import NearestSensorImputer, SupportedLastImputer
from app.missing_values_percentage_filter import MissingValuesPercentageFilter
```

[ ]:
```
ANALYZED_VARIABLE = constants.PM10

SENSOR_METADATA = "input_files/sensor_metadata.parquet"
MEASUREMENTS_24H = "input_files/measurements_24h.parquet"

OUTPUT_FILE_NEAREST = f"input_files/
 ↪nearest_imputed_measurements_24h_{ANALYZED_VARIABLE}.parquet"
OUTPUT_FILE_LAST = f"input_files/
 ↪last_imputed_measurements_24h_{ANALYZED_VARIABLE}.parquet"

MIN_YEAR = 2017
MAX_YEAR = 2023
```

[ ]:
```
sensor_metadata = pd.read_parquet(SENSOR_METADATA)
raw_measurements = pd.read_parquet(MEASUREMENTS_24H)
```

[ ]:
```
measurements = raw_measurements[
    [constants.TIMESTAMP_COLUMN, constants.UNIQUE_ID, ANALYZED_VARIABLE]
]
measurements = measurements.query(
    f"{constants.TIMESTAMP_COLUMN}.dt.year >= {MIN_YEAR} and "
    f"{constants.TIMESTAMP_COLUMN}.dt.year <= {MAX_YEAR}"
)
measurements.shape
```

```
[ ]: valid_curves = MissingValuesPercentageFilter(threshold=0.05).fit_transform(
         measurements, ANALYZED_VARIABLE
     )

     filtered_measurements = measurements[
         (measurements[constants.UNIQUE_ID].isin(valid_curves[constants.UNIQUE_ID].
      ↪unique()))
         & (measurements[constants.TIMESTAMP_COLUMN].dt.year.
      ↪isin(valid_curves[constants.YEAR].unique()))
     ]
     filtered_measurements.shape
```

0.1 Imputacja na podstawie najbliższego sensora

```
[ ]: loc_sensor = sensor_metadata[[constants.SENSOR_ID, constants.LATITUDE,␣
      ↪constants.LONGITUDE]]
     gdf_loc_sensor = gpd.GeoDataFrame(
         loc_sensor,
         geometry=gpd.points_from_xy(loc_sensor[constants.LONGITUDE],␣
      ↪loc_sensor[constants.LATITUDE]),
         crs=constants.GLOBAL_EPSG,
     )
     gdf_loc_sensor = gdf_loc_sensor.to_crs(constants.POLAND_EPSG)
     distance_matrix = gdf_loc_sensor.geometry.apply(
         lambda x: gdf_loc_sensor.distance(x)
     ).values.tolist()
```

```
[ ]: nearest_sensor_imputer = NearestSensorImputer(
         distance_matrix=distance_matrix, sensor_ids=gdf_loc_sensor[constants.
      ↪SENSOR_ID].values.tolist()
     )
     nearest_sensor_imputed_measurements = nearest_sensor_imputer.
      ↪fit_transform(X=filtered_measurements)
     nearest_sensor_imputed_measurements.to_parquet(OUTPUT_FILE_NEAREST, index=False)
```

0.2 Imputacja brakujących wartości na podstawie ostatniej znanej wartośći z uwzględnieniem najbliższych sensorów

```
[ ]: supported_last_imputer =␣
      ↪SupportedLastImputer(support_imputer=nearest_sensor_imputer)
     supported_last_imputed_measurements = supported_last_imputer.
      ↪fit_transform(X=filtered_measurements)
     supported_last_imputed_measurements.to_parquet(OUTPUT_FILE_LAST, index=False)
```

```
[ ]: nearest_sensor_imputed_measurements = pd.read_parquet(OUTPUT_FILE_NEAREST)
```

```
supported_last_imputed_measurements = pd.read_parquet(OUTPUT_FILE_LAST).
  ↪drop_duplicates()
```

[ ]:
```
nearest_sensor_imputed_measurements.isna().sum().sum(),␣
  ↪nearest_sensor_imputed_measurements.shape
```

[ ]:
```
supported_last_imputed_measurements.isna().sum().sum(),␣
  ↪supported_last_imputed_measurements.shape
```

[ ]:
```
supported_last_imputed_measurements.to_parquet(OUTPUT_FILE_LAST, index=False)
```