DSC680_WK07_Kim-Schreck

July 21, 2024

0.1 DSC680_project_02_Ross-Kim-Schreck

1 milestone 01

[1]:
```
# imports

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from datetime import datetime
import time
import numpy as np
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from pandas import read_csv
from pandas import DataFrame
from statsmodels.tsa.arima.model import ARIMA
from matplotlib import pyplot
from statsmodels.tsa.stattools import adfuller
import plotly.express as px
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score, RepeatedStratifiedKFold
from numpy import mean, std
import xgboost as xgb
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
```

```
import sys
import time
import traceback
from joblib.externals.loky import set_loky_pickler
'''from joblib import parallel_config'''
from joblib import Parallel, delayed
from joblib import wrap_non_picklable_objects
import multiprocessing as mp
```

[2]:
```
# 06.02.01.01
# read csv
# assign variable
# dt01

dt01_Air_Quality_Seoul_______00 = pd.read_csv('Air_Quality_Seoul.csv')
```

[3]:
```
# 06.02.01.02
# return first and last ten rows
# dt01

print(dt01_Air_Quality_Seoul_______00.head(10))
print(dt01_Air_Quality_Seoul_______00.tail(10))
```

```
      Measurement date  Station code    SO2    NO2     O3   CO  PM10  PM2.5
0  2017-01-01 00:00:00           101  0.004  0.059  0.002  1.2  73.0   57.0
1  2017-01-01 01:00:00           101  0.004  0.058  0.002  1.2  71.0   59.0
2  2017-01-01 02:00:00           101  0.004  0.056  0.002  1.2  70.0   59.0
3  2017-01-01 03:00:00           101  0.004  0.056  0.002  1.2  70.0   58.0
4  2017-01-01 04:00:00           101  0.003  0.051  0.002  1.2  69.0   61.0
5  2017-01-01 05:00:00           101  0.003  0.046  0.002  1.1  70.0   61.0
6  2017-01-01 06:00:00           101  0.003  0.049  0.002  1.1  66.0   57.0
7  2017-01-01 07:00:00           101  0.003  0.045  0.002  1.0  71.0   60.0
8  2017-01-01 08:00:00           101  0.004  0.047  0.002  1.1  72.0   60.0
9  2017-01-01 09:00:00           101  0.003  0.047  0.002  1.1  74.0   63.0
           Measurement date  Station code    SO2  …   CO  PM10  PM2.5
866449  2020-12-31 14:00:00           125  0.003  …  0.4  33.0   20.0
866450  2020-12-31 15:00:00           125  0.003  …  0.4  31.0   14.0
866451  2020-12-31 16:00:00           125  0.003  …  0.4  34.0   19.0
866452  2020-12-31 17:00:00           125  0.002  …  0.5  34.0   18.0
866453  2020-12-31 18:00:00           125  0.003  …  0.5  40.0   21.0
866454  2020-12-31 19:00:00           125  0.003  …  0.5  35.0   19.0
866455  2020-12-31 20:00:00           125  0.003  …  0.5  34.0   20.0
866456  2020-12-31 21:00:00           125  0.003  …  0.5  33.0   18.0
866457  2020-12-31 22:00:00           125  0.003  …  0.4  35.0   18.0
866458  2020-12-31 23:00:00           125  0.002  …  0.4  25.0   15.0

[10 rows x 8 columns]
```

[4]:
```
# 06.02.01.03
# return dimensions
# dt01

print(dt01_Air_Quality_Seoul_______00.shape)
```

```
(866459, 8)
```

[5]:
```
# 06.02.01.04
# confirm column names
# dt01

dt01_Air_Quality_Seoul_______00.columns
```

[5]:
```
Index(['Measurement date', 'Station code', 'SO2', 'NO2', 'O3', 'CO', 'PM10',
       'PM2.5'],
      dtype='object')
```

[6]:
```
# 06.02.01.05
# column rename to remove spaces
# dt01

dt01_Air_Quality_Seoul_______00_rn = dt01_Air_Quality_Seoul_______00.
 ↪rename(columns = {
    'Measurement date': 'date_measure',
    'Station code': 'code_station',
    'SO2': 'SO2',
    'NO2': 'NO2',
    'O3': 'O3',
    'CO': 'CO',
    'PM10': 'PM10',
    'PM2.5': 'PM2_5'
    })
```

[7]:
```
# 06.02.01.06
# confirm column names
# dt01

dt01_Air_Quality_Seoul_______00_rn.columns
```

[7]:
```
Index(['date_measure', 'code_station', 'SO2', 'NO2', 'O3', 'CO', 'PM10',
       'PM2_5'],
      dtype='object')
```

[8]:
```
# 06.02.02.01
# change column to datetime
# dt01
```

```
ts, ms = '20.12.2016 09:38:42,76'.split(',')
dt01_Air_Quality_Seoul_____00_rn['date_measure'] = datetime.strptime(ts, '%d.
 ↪%m.%Y %H:%M:%S')
dt01_Air_Quality_Seoul_____00_rn_cv = dt01_Air_Quality_Seoul_____00_rn
```

[9]:
```
# 06.02.02.02
# change column to integer
# dt01

dt01_Air_Quality_Seoul_____00_rn_cv['code_station'] =␣
 ↪dt01_Air_Quality_Seoul_____00_rn['code_station'].astype(int)
```

[10]:
```
# 06.02.02.03
# change column to float
# dt01

dt01_Air_Quality_Seoul_____00_rn_cv['SO2'] =␣
 ↪dt01_Air_Quality_Seoul_____00_rn['SO2'].astype(float)
```

[11]:
```
# 06.02.02.04
# change column to float
# dt01

dt01_Air_Quality_Seoul_____00_rn_cv['NO2'] =␣
 ↪dt01_Air_Quality_Seoul_____00_rn['NO2'].astype(float)
```

[12]:
```
# 06.02.02.05
# change column to float
# dt01

dt01_Air_Quality_Seoul_____00_rn_cv['03'] =␣
 ↪dt01_Air_Quality_Seoul_____00_rn['03'].astype(float)
```

[13]:
```
# 06.02.02.06
# change column to float
# dt01

dt01_Air_Quality_Seoul_____00_rn_cv['CO'] =␣
 ↪dt01_Air_Quality_Seoul_____00_rn['CO'].astype(float)
```

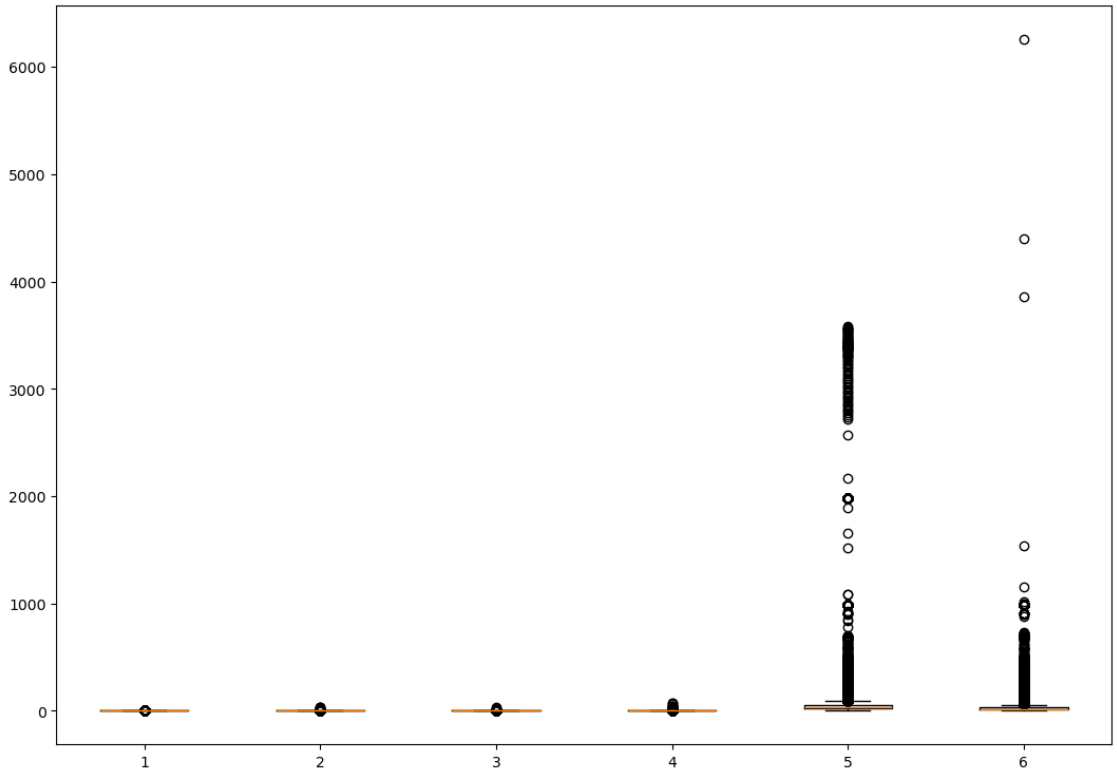
[14]:
```
# 06.02.02.07
# change column to float
# dt01

dt01_Air_Quality_Seoul_____00_rn_cv['PM10'] =␣
 ↪dt01_Air_Quality_Seoul_____00_rn['PM10'].astype(float)
```

[15]:
```
# 06.02.02.08
# change column to float
# dt01

dt01_Air_Quality_Seoul_______00_rn_cv['PM2_5'] =␣
 ↪dt01_Air_Quality_Seoul_______00_rn['PM2_5'].astype(float)
```

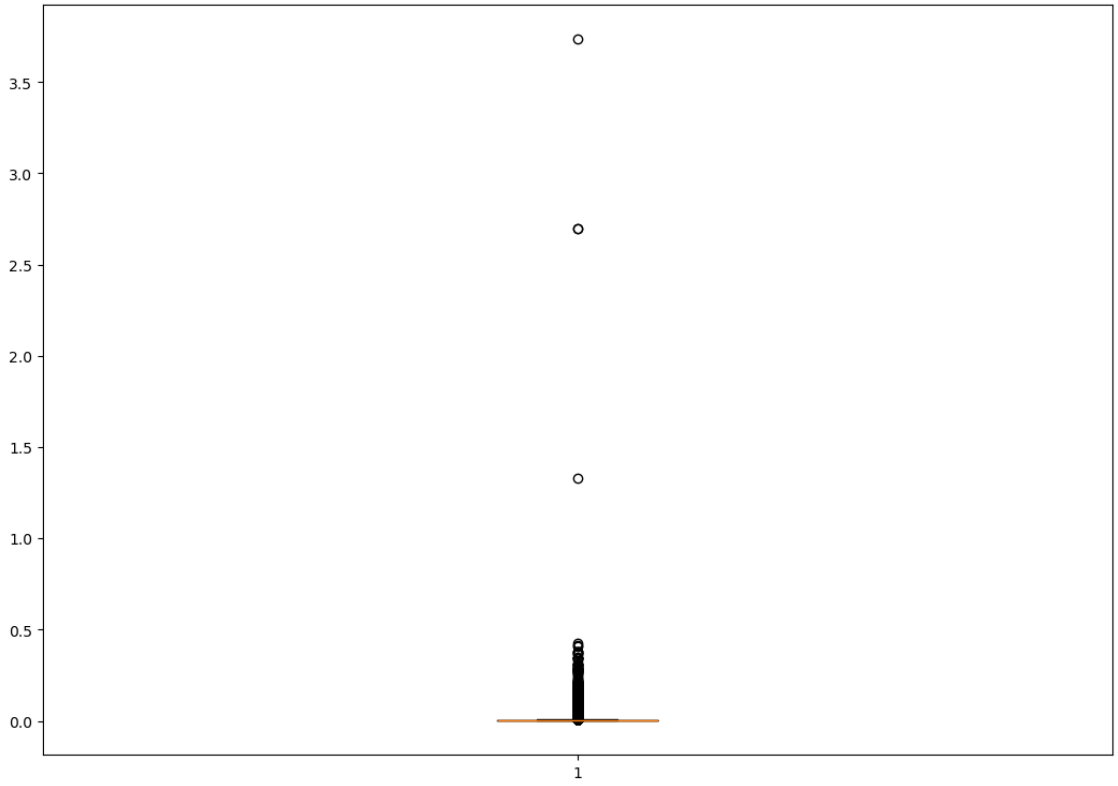
[16]:
```
# 06.02.02.09
# confirm datatype
# dt01

print(dt01_Air_Quality_Seoul_______00_rn_cv['date_measure'].dtypes)
print(dt01_Air_Quality_Seoul_______00_rn_cv['code_station'].dtypes)
print(dt01_Air_Quality_Seoul_______00_rn_cv['SO2'].dtypes)
print(dt01_Air_Quality_Seoul_______00_rn_cv['NO2'].dtypes)
print(dt01_Air_Quality_Seoul_______00_rn_cv['O3'].dtypes)
print(dt01_Air_Quality_Seoul_______00_rn_cv['CO'].dtypes)
print(dt01_Air_Quality_Seoul_______00_rn_cv['PM10'].dtypes)
print(dt01_Air_Quality_Seoul_______00_rn_cv['PM2_5'].dtypes)
```

```
datetime64[us]
int64
float64
float64
float64
float64
float64
float64
```

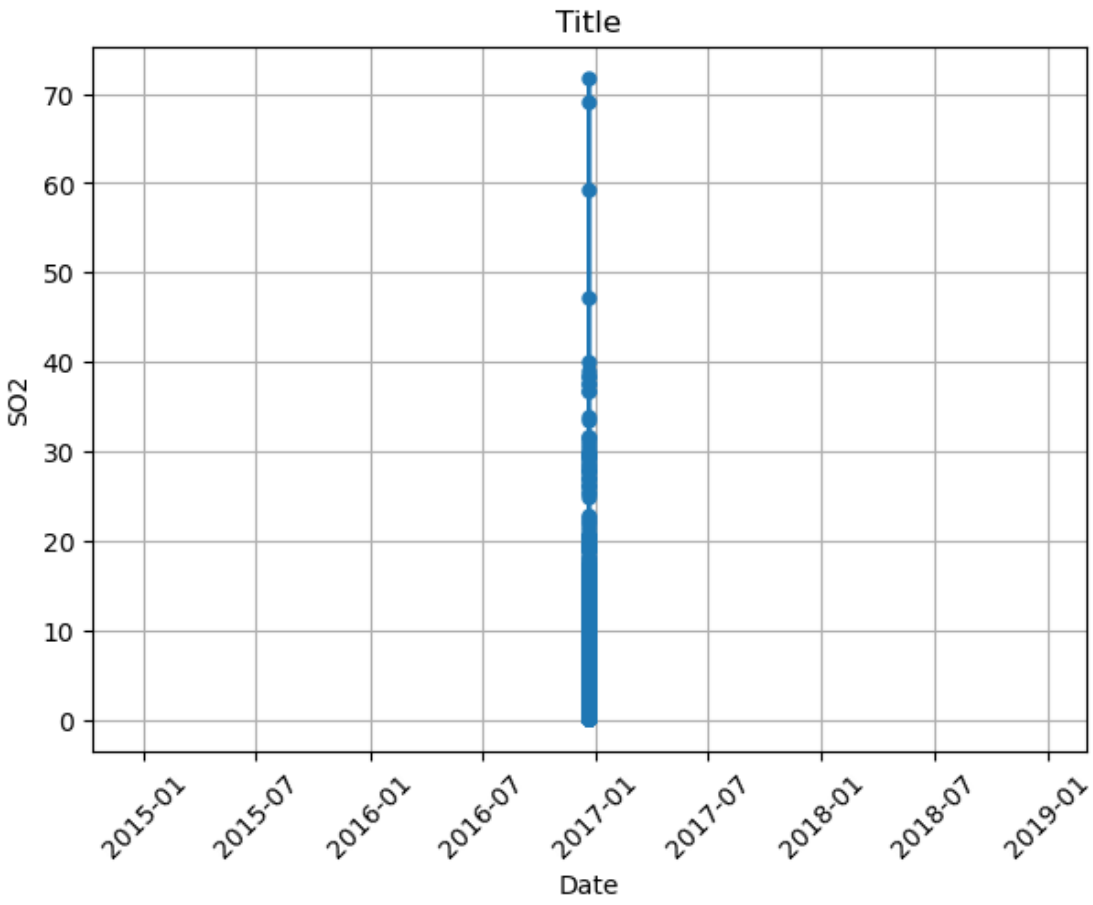
[17]:
```
# 06.02.03.01
# boxplot
# dt01

np.random.seed(10)
dt01_Air_Quality_Seoul_______00_rn_cv_boxplot =␣
 ↪[dt01_Air_Quality_Seoul_______00_rn_cv['SO2'],␣
 ↪dt01_Air_Quality_Seoul_______00_rn_cv['NO2'],␣
 ↪dt01_Air_Quality_Seoul_______00_rn_cv['O3'],␣
 ↪dt01_Air_Quality_Seoul_______00_rn_cv['CO'],␣
 ↪dt01_Air_Quality_Seoul_______00_rn_cv['PM10'],␣
 ↪dt01_Air_Quality_Seoul_______00_rn_cv['PM2_5']]
fig = plt.figure(figsize =(10, 7))
ax = fig.add_axes([0, 0, 1, 1])
bp = ax.boxplot(dt01_Air_Quality_Seoul_______00_rn_cv_boxplot)
plt.show()
```



[18]:
```
# 06.02.03.02
# boxplot
# dt01

np.random.seed(10)
dt01_Air_Quality_Seoul_______00_rn_cv_boxplot =␣
 ↪[dt01_Air_Quality_Seoul_______00_rn_cv['SO2']]
fig = plt.figure(figsize =(10, 7))
ax = fig.add_axes([0, 0, 1, 1])
bp = ax.boxplot(dt01_Air_Quality_Seoul_______00_rn_cv_boxplot)
plt.show()
```



[19]:
```
# 06.02.03.03
# plot
# dt01

plt.figure(figsize=(7, 5))
plt.plot(dt01_Air_Quality_Seoul_______00_rn_cv['date_measure'],␣
 ↪dt01_Air_Quality_Seoul_______00_rn_cv['CO'], marker='o', linestyle='-',␣
 ↪markersize=5)
plt.xticks(rotation = 45)
plt.xlabel('Date')
plt.ylabel('SO2')
plt.title('Title')
plt.grid(True)
plt.show()
```



[20]:
```
# 06.02.03.04
# render plot
# dt01

'''plt.figure(figsize=(12,8))
plt.bar(dt01_Air_Quality_Seoul_______00_rn_cv['code_station'],␣
 ↪dt01_Air_Quality_Seoul_______00_rn_cv['SO2'], color = '#890343',
       width = 0.4)
plt.xticks(fontsize = 10)
plt.xticks(rotation = 45)
plt.yticks(fontsize = 6)
plt.yticks(rotation = 0)
plt.xlabel('Date / Time')
plt.ylabel('SO2')
plt.title('Title')
plt.show()'''
```

[20]:
```
"plt.figure(figsize=(12,8))\nplt.bar(dt01_Air_Quality_Seoul_______00_rn_cv['code
_station'], dt01_Air_Quality_Seoul_______00_rn_cv['SO2'], color = '#890343', \n
width = 0.4)\nplt.xticks(fontsize = 10)\nplt.xticks(rotation =
45)\nplt.yticks(fontsize = 6)\nplt.yticks(rotation = 0)\nplt.xlabel('Date /
Time')\nplt.ylabel('SO2')\nplt.title('Title')\nplt.show()"
```

[21]:
```
# 06.02.03.05
# render plot
# dt01

'''dt01_x00 = dt01_Air_Quality_Seoul_______00_rn['date_measure']
dt01_y01 = dt01_Air_Quality_Seoul_______00_rn['code_station']
dt01_y02 = dt01_Air_Quality_Seoul_______00_rn['SO2']
dt01_y03 = dt01_Air_Quality_Seoul_______00_rn['NO2']
dt01_y04 = dt01_Air_Quality_Seoul_______00_rn['03']
dt01_y05 = dt01_Air_Quality_Seoul_______00_rn['CO']
dt01_y06 = dt01_Air_Quality_Seoul_______00_rn['PM10']
dt01_y07 = dt01_Air_Quality_Seoul_______00_rn['PM2_5']
plt.figure(figsize=(18,8))
plt.plot(dt01_x00, dt01_y01, label = "code_station", linestyle="-")
plt.plot(dt01_x00, dt01_y02, label = "SO2", linestyle="-")
plt.plot(dt01_x00, dt01_y03, label = "NO2", linestyle="-")
plt.plot(dt01_x00, dt01_y04, label = "03", linestyle="-")
plt.plot(dt01_x00, dt01_y05, label = "CO", linestyle="-")
plt.plot(dt01_x00, dt01_y06, label = "PM10", linestyle="-")
plt.plot(dt01_x00, dt01_y07, label = "PM2_5", linestyle="-")
plt.legend()
plt.show()'''
```

[21]:
```
'dt01_x00 = dt01_Air_Quality_Seoul_______00_rn[\'date_measure\']\ndt01_y01 =
dt01_Air_Quality_Seoul_______00_rn[\'code_station\']\ndt01_y02 =
dt01_Air_Quality_Seoul_______00_rn[\'SO2\']\ndt01_y03 =
dt01_Air_Quality_Seoul_______00_rn[\'NO2\']\ndt01_y04 =
dt01_Air_Quality_Seoul_______00_rn[\'03\']\ndt01_y05 =
dt01_Air_Quality_Seoul_______00_rn[\'CO\']\ndt01_y06 =
dt01_Air_Quality_Seoul_______00_rn[\'PM10\']\ndt01_y07 = dt01_Air_Quality_Seoul_
______00_rn[\'PM2_5\']\nplt.figure(figsize=(18,8))\nplt.plot(dt01_x00, dt01_y01,
label = "code_station", linestyle="-")\nplt.plot(dt01_x00, dt01_y02, label =
"SO2", linestyle="-")\nplt.plot(dt01_x00, dt01_y03, label = "NO2",
linestyle="-")\nplt.plot(dt01_x00, dt01_y04, label = "03",
linestyle="-")\nplt.plot(dt01_x00, dt01_y05, label = "CO",
linestyle="-")\nplt.plot(dt01_x00, dt01_y06, label = "PM10",
linestyle="-")\nplt.plot(dt01_x00, dt01_y07, label = "PM2_5",
linestyle="-")\nplt.legend() \nplt.show()'
```

[22]:
```
# 06.02.04.01
# read csv
```

```
# assign variable
# dt02

dt02_Air_Quality_cities______00 = pd.
 ↪read_csv('cities_air_quality_water_pollution.18-10-2021.csv')
```

[23]:
```
# 06.02.04.02
# return first and last ten rows
# dt02

print(dt02_Air_Quality_cities______00.head(10))
print(dt02_Air_Quality_cities______00.tail(10))
```

```
              City                  "Region"  …  "AirQuality"
"WaterPollution"
0     New York City               "New York"  …     46.816038
49.504950
1  Washington, D.C.   "District of Columbia"  …     66.129032
49.107143
2     San Francisco             "California"  …     60.514019
43.000000
3            Berlin                       ""  …     62.364130
28.612717
4       Los Angeles             "California"  …     36.621622
61.299435
5              Bern         "Canton of Bern"  …     94.318182
12.500000
6            Geneva       "Canton of Geneva"  …     71.538462
17.372881
7            Zurich       "Canton of Zurich"  …     83.809524
10.714286
8             Basel                       ""  …     81.666667
26.923077
9            London                "England"  …     37.042254
40.716374

[10 rows x 5 columns]
             City              "Region"  …  "AirQuality"   "WaterPollution"
3953       Teruel              "Aragon"  …    100.000000          50.000000
3954   Piracicaba           "Sao Paulo"  …     40.000000          25.000000
3955  Ciudad Real   "Castile-La Mancha"  …     82.142857           0.000000
3956     Palencia    "Castile and Leon"  …     79.166667          37.500000
3957       Jubail    "Eastern Province"  …     30.468750          38.793103
3958        Yanbu     "Medina Province"  …      0.000000          50.000000
3959      Cordoba           "Andalusia"  …     85.714286           8.333333
3960          Vic           "Catalonia"  …    100.000000           0.000000
3961      Segovia    "Castile and Leon"  …    100.000000           0.000000
3962  Zamora city    "Castile and Leon"  …    100.000000          50.000000
```

```
[10 rows x 5 columns]
```

[24]:
```
# 06.02.04.03
# return dimensions
# dt02

print(dt02_Air_Quality_cities______00.shape)
```

```
(3963, 5)
```

[25]:
```
# 06.02.04.04
# confirm column names
# dt02

dt02_Air_Quality_cities______00.columns
```

[25]:
```
Index(['City', ' "Region"', ' "Country"', ' "AirQuality"',
       ' "WaterPollution"'],
      dtype='object')
```

[26]:
```
# 06.02.04.05
# column rename to remove spaces and quotes
# dt02

dt02_Air_Quality_cities______00_rn = dt02_Air_Quality_cities______00.
 ↪rename(columns = {
    'City': 'city',
    ' "Region"': 'region',
    ' "Country"': 'country',
    ' "AirQuality"': 'air_quality',
    ' "WaterPollution"': 'water_pollution'
    })
```

[27]:
```
# 06.02.04.06
# confirm column names
# dt02

dt02_Air_Quality_cities______00_rn.columns
```

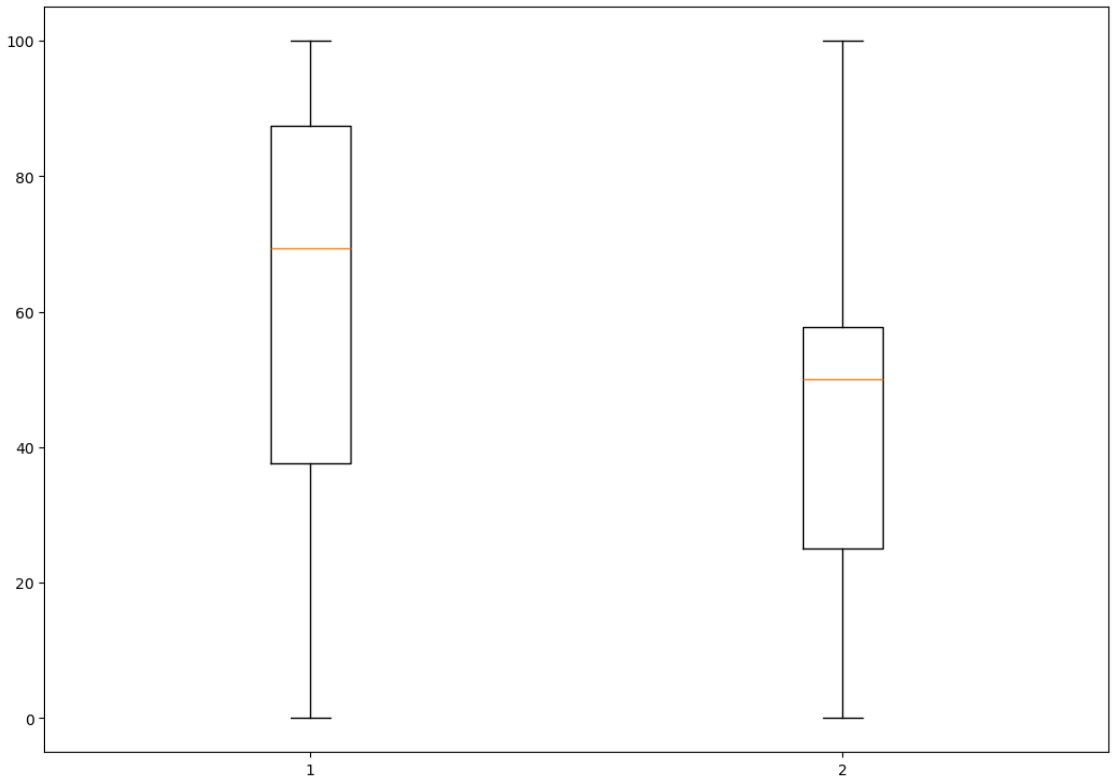
[27]:
```
Index(['city', 'region', 'country', 'air_quality', 'water_pollution'],
dtype='object')
```

[28]:
```
# 06.02.04.07
# confirm types
# dt02

print(dt02_Air_Quality_cities______00_rn['city'].dtypes)
```

```
print(dt02_Air_Quality_cities______00_rn['region'].dtypes)
print(dt02_Air_Quality_cities______00_rn['country'].dtypes)
print(dt02_Air_Quality_cities______00_rn['air_quality'].dtypes)
print(dt02_Air_Quality_cities______00_rn['water_pollution'].dtypes)
```

```
object
object
object
float64
float64
```

[29]:
```
# 06.02.05.01
# boxplot
# dt02

np.random.seed(10)
dt02_Air_Quality_cities______00_rn_boxplot =␣
 ↪[dt02_Air_Quality_cities______00_rn['air_quality'],␣
 ↪dt02_Air_Quality_cities______00_rn['water_pollution']]
fig = plt.figure(figsize =(10, 7))
ax = fig.add_axes([0, 0, 1, 1])
bp = ax.boxplot(dt02_Air_Quality_cities______00_rn_boxplot)
plt.show()
```



[30]:
```
# 06.02.05.02
# render plot
# dt02

'''dt02_x00 = dt02_Air_Quality_cities_____00_rn['city']
dt02_y01 = dt02_Air_Quality_cities_____00_rn['region']
dt02_y02 = dt02_Air_Quality_cities_____00_rn['country']
dt02_y03 = dt02_Air_Quality_cities_____00_rn['air_quality']
dt02_y04 = dt02_Air_Quality_cities_____00_rn['water_pollution']
plt.figure(figsize=(18,8))
plt.plot(dt02_x00, dt02_y03, label = "air_quality", linestyle="-")
plt.plot(dt02_x00, dt02_y04, label = "water_pollution", linestyle="-")
plt.legend()
plt.show()'''
```

[30]:
```
'dt02_x00 = dt02_Air_Quality_cities_____00_rn[\'city\']\ndt02_y01 =
dt02_Air_Quality_cities_____00_rn[\'region\']\ndt02_y02 =
dt02_Air_Quality_cities_____00_rn[\'country\']\ndt02_y03 =
dt02_Air_Quality_cities_____00_rn[\'air_quality\']\ndt02_y04 = dt02_Air_Quality
_cities_____00_rn[\'water_pollution\']\nplt.figure(figsize=(18,8))\nplt.plot(dt
02_x00, dt02_y03, label = "air_quality", linestyle="-")\nplt.plot(dt02_x00,
dt02_y04, label = "water_pollution", linestyle="-")\nplt.legend() \nplt.show()'
```

[31]:
```
# 06.02.05.03
# render plot
# dt02

'''plt.figure(figsize=(18,8))
plt.bar(dt02_Air_Quality_cities_____00_rn['city'],␣
 ↪dt02_Air_Quality_cities_____00_rn['air_quality'], color = '#890343',
       width = 0.4)
plt.xticks(fontsize = 4)
plt.xticks(rotation = 45)
plt.yticks(fontsize = 6)
plt.yticks(rotation = 0)
plt.xlabel('City')
plt.ylabel('Air Quality')
plt.title('Air Quality')
plt.show()'''
```

[31]:
```
"plt.figure(figsize=(18,8))\nplt.bar(dt02_Air_Quality_cities_____00_rn['city'],
dt02_Air_Quality_cities_____00_rn['air_quality'], color = '#890343', \n
width = 0.4)\nplt.xticks(fontsize = 4)\nplt.xticks(rotation =
45)\nplt.yticks(fontsize = 6)\nplt.yticks(rotation =
0)\nplt.xlabel('City')\nplt.ylabel('Air Quality')\nplt.title('Air
Quality')\nplt.show()"
```

[32]:
```
# 06.02.06.01
# preparing data for modeling
# add index column
# index column to select specific rows
# dt02

dt02_Air_Quality_cities_____00_rn.insert(0, 'index', range(0, 0 +␣
 ↪len(dt02_Air_Quality_cities_____00_rn)))
```

[33]:
```
# 06.02.06.02
# preparing data for modeling
# create dummy variables
# due to returning boolean values, converting dummmies to integers
# dt02

dt02_Air_Quality_cities_____00_rn_dv = pd.
 ↪get_dummies(dt02_Air_Quality_cities_____00_rn, drop_first = True, dtype =␣
 ↪int)
```

[34]:
```
# 06.02.06.03
# preparing data for modeling
# split data
# select columns
# dt02

dt02_x01 = dt02_Air_Quality_cities_____00_rn_dv.drop(['air_quality'], axis = 1)
dt02_y01 = dt02_Air_Quality_cities_____00_rn_dv['air_quality']
```

[35]:
```
# 06.02.06.04
# preparing data for modeling
# split into train and test
# dt02

dt02_x01_trn, dt02_x01_tst, dt02_y01_trn, dt02_y01_tst =␣
 ↪train_test_split(dt02_x01, dt02_y01, test_size = 0.3, random_state = 0)
```

[36]:
```
# 06.02.06.05
# preparing data for modeling
# assign regression variable
# dt02

dt02_lr01 = LinearRegression()
```

[37]:
```
# 06.02.06.06
# fit data for modeling
# fit variables to model
# dt02
```

```
dt02_lr01.fit(dt02_x01_trn, dt02_y01_trn)
```

[37]:
```
LinearRegression()
```

[38]:
```
# 06.02.06.07
# predict data for modeling
# fit variables to model
# dt02

dt02_y01_pdct = dt02_lr01.predict(dt02_x01_tst)
```

[39]:
```
# 06.02.06.08
# preparing data for modeling
# assign variable for rmse and r2
# dt02

dt02_rmse01 = np.sqrt(mean_squared_error(dt02_y01_tst, dt02_y01_pdct))
dt02_r201 = r2_score(dt02_y01_tst, dt02_y01_pdct)
```

[40]:
```
# 06.02.06.09
# run model
# return rmse and r2 dt02
# rmse: 28864053.155290764
# r2: -851681671710.9026
# dt02

print(f'rmse: {dt02_rmse01}')
print(f'r2: {dt02_r201}')
```

```
rmse: 51165659.245247684
r2: -2676207714892.1455
```

[41]:
```
# 06.02.06.10
# assign variable for pca
# dt02

pca = PCA(.9)
```

[42]:
```
# 06.02.06.11
# calculate pca
# dt02

pca.fit(dt02_x01_trn)
dt02_x01_pca_trn = pca.transform(dt02_x01_trn)
dt02_x01_pca_tst = pca.transform(dt02_x01_tst)
```

[43]:
```
# 06.02.06.12
# return pca calculation matrix
# dt02

print(f'features in pca matrix: {dt02_x01_pca_trn.shape[1]}')
```

```
features in pca matrix: 1
```

[44]:
```
# 06.02.07.01
# preparing data for modeling
# add index column
# index column to select specific rows
# dt02

'''dt02_Air_Quality_cities_____00_rn.insert(0, 'index', range(0, 0 +␣
 ↪len(dt02_Air_Quality_cities_____00_rn)))'''
```

[44]:
```
"dt02_Air_Quality_cities_____00_rn.insert(0, 'index', range(0, 0 +
len(dt02_Air_Quality_cities_____00_rn)))"
```

[45]:
```
# 06.02.07.02
# preparing data for modeling
# create dummy variables
# due to returning boolean values, converting dummmies to integers
# dt02

dt02_Air_Quality_cities_____00_rn_dv = pd.
 ↪get_dummies(dt02_Air_Quality_cities_____00_rn, drop_first = True, dtype =␣
 ↪int)
```

[46]:
```
# 06.02.07.03
# preparing data for modeling
# split data
# select columns
# dt02

dt02_x02 = dt02_Air_Quality_cities_____00_rn_dv.drop(['water_pollution'], axis␣
 ↪= 1)
dt02_y02 = dt02_Air_Quality_cities_____00_rn_dv['water_pollution']
```

[47]:
```
# 06.02.07.04
# preparing data for modeling
# split into train and test
# dt02

dt02_x02_trn, dt02_x02_tst, dt02_y02_trn, dt02_y02_tst =␣
 ↪train_test_split(dt02_x02, dt02_y02, test_size = 0.3, random_state = 0)
```

[48]:
```
# 06.02.07.05
# preparing data for modeling
# assign regression variable
# dt02

dt02_lr02 = LinearRegression()
```

[49]:
```
# 06.02.07.06
# fit data for modeling
# fit variables to model
# dt02

dt02_lr02.fit(dt02_x02_trn, dt02_y02_trn)
```

[49]:
```
LinearRegression()
```

[50]:
```
# 06.02.07.07
# predict data for modeling
# fit variables to model
# dt02

dt02_y02_pdct = dt02_lr02.predict(dt02_x02_tst)
```

[51]:
```
# 06.02.07.08
# preparing data for modeling
# assign variable for rmse and r2
# dt02

dt02_rmse02 = np.sqrt(mean_squared_error(dt02_y02_tst, dt02_y02_pdct))
dt02_r202 = r2_score(dt02_y02_tst, dt02_y02_pdct)
```

[52]:
```
# 06.02.07.09
# run model
# return rmse and r2 dt02
# rmse: 191190784.13206095
# r2: -52543986717271.29
# dt02

print(f'rmse: {dt02_rmse02}')
print(f'r2: {dt02_r202}')
```

```
rmse: 37731552.29782568
r2: -2046437370417.3125
```

[53]:
```
# 06.02.07.10
# assign variable for pca
# dt02

```

```
pca = PCA(.9)
```

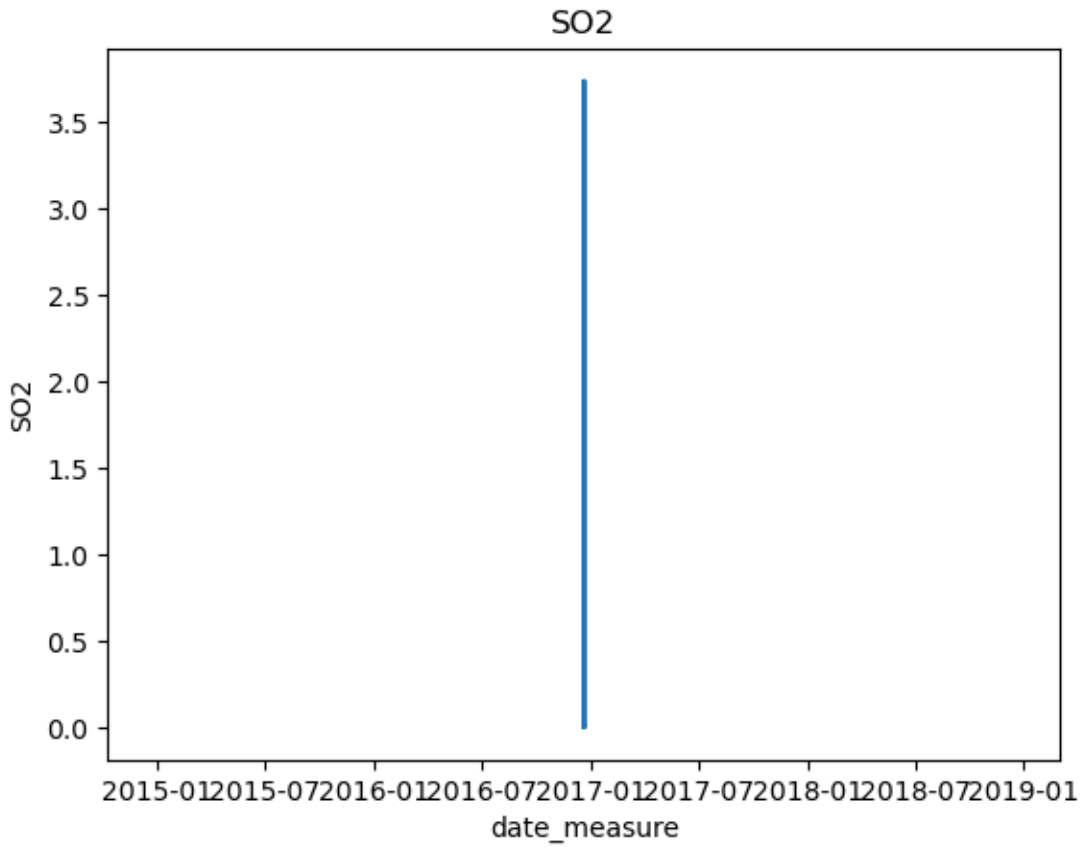
[54]:
```
# 06.02.07.11
# calculate pca
# dt02

pca.fit(dt02_x02_trn)
dt02_x02_pca_trn = pca.transform(dt02_x02_trn)
dt02_x02_pca_tst = pca.transform(dt02_x02_tst)
```

[55]:
```
# 06.02.07.12
# return pca calculation matrix
# dt02

print(f'features in pca matrix: {dt02_x02_pca_trn.shape[1]}')
```

```
features in pca matrix: 1
```

[56]:
```
# 06.02.08.01
# load dataset
# dt01

dt01_Air_Quality_Seoul_______00_rn_cv['date_measure'] = pd.
 ↪to_datetime(dt01_Air_Quality_Seoul_______00_rn_cv['date_measure'])
dt01_Air_Quality_Seoul_______00_rn_cv.set_index('date_measure', inplace=True)
```

[57]:
```
# 06.02.08.02
# plot time series
# dt01

plt.plot(dt01_Air_Quality_Seoul_______00_rn_cv['SO2'])
plt.title('SO2')
plt.xlabel('date_measure')
plt.ylabel('SO2')
plt.show()
```



2 milestone 02

[58]:
```
# 06.02.09.01
# read csv
# assign variable
# dt03

dt03_measurement_info________00 = pd.read_csv('Measurement_info.csv')
```

[59]:
```
# 06.02.09.02
# read csv
# assign variable
# dt04

dt04_measurement_info_item___00 = pd.read_csv('Measurement_item_info.csv')
```

[60]:
```
# 06.02.09.03
# read csv
# assign variable
```

```
# dt05

dt05_measurement_info_stn____00 = pd.read_csv('Measurement_station_info.csv')
```

[61]:
```
# 06.02.09.04
# return first and last ten rows
# dt03

print(dt03_measurement_info________00.head(10))
print(dt03_measurement_info________00.tail(10))
```

```
   Measurement date  Station code  Item code  Average value  Instrument status
0  2017-01-01 00:00           101          1          0.004                  0
1  2017-01-01 00:00           101          3          0.059                  0
2  2017-01-01 00:00           101          5          1.200                  0
3  2017-01-01 00:00           101          6          0.002                  0
4  2017-01-01 00:00           101          8         73.000                  0
5  2017-01-01 00:00           101          9         57.000                  0
6  2017-01-01 00:00           102          1          0.006                  0
7  2017-01-01 00:00           102          3          0.068                  0
8  2017-01-01 00:00           102          5          1.300                  0
9  2017-01-01 00:00           102          6          0.002                  0
         Measurement date  Station code  …  Average value  Instrument status
3885056  2019-12-31 23:00           113  …         11.000                  0
3885057  2019-12-31 23:00           122  …         15.000                  0
3885058  2019-12-31 23:00           124  …         13.000                  0
3885059  2019-12-31 23:00           124  …          0.500                  0
3885060  2019-12-31 23:00           113  …          0.002                  0
3885061  2019-12-31 23:00           123  …         13.000                  0
3885062  2019-12-31 23:00           118  …         24.000                  0
3885063  2019-12-31 23:00           105  …         19.000                  0
3885064  2019-12-31 23:00           125  …          0.037                  0
3885065  2019-12-31 23:00           108  …          0.030                  0

[10 rows x 5 columns]
```

[62]:
```
# 06.02.09.05
# return first and last ten rows
# dt04

print(dt04_measurement_info_item___00.head(10))
print(dt04_measurement_info_item___00.tail(10))
```

```
   Item code Item name  … Bad(Yellow)  Very bad(Red)
0          1       SO2  …         0.15            1.0
1          3       NO2  …         0.20            2.0
2          5        CO  …        15.00           50.0
3          6        O3  …         0.15            0.5
```

```
4          8      PM10  …      150.00          600.0
5          9     PM2.5  …       75.00          500.0

[6 rows x 7 columns]
   Item code Item name  … Bad(Yellow)  Very bad(Red)
0          1       SO2  …        0.15            1.0
1          3       NO2  …        0.20            2.0
2          5        CO  …       15.00           50.0
3          6        O3  …        0.15            0.5
4          8      PM10  …      150.00          600.0
5          9     PM2.5  …       75.00          500.0

[6 rows x 7 columns]
```

[63]:
```
# 06.02.09.06
# return first and last ten rows
# dt05

print(dt05_measurement_info_stn____00.head(10))
print(dt05_measurement_info_stn____00.tail(10))
```

```
   Station code Station name(district)  …   Latitude   Longitude
0           101              Jongno-gu  …  37.572016  127.005008
1           102                Jung-gu  …  37.564263  126.974676
2           103             Yongsan-gu  …  37.540033  127.004850
3           104           Eunpyeong-gu  …  37.609823  126.934848
4           105           Seodaemun-gu  …  37.593742  126.949679
5           106                Mapo-gu  …  37.555580  126.905597
6           107           Seongdong-gu  …  37.541864  127.049659
7           108            Gwangjin-gu  …  37.547180  127.092493
8           109          Dongdaemun-gu  …  37.575743  127.028885
9           110            Jungnang-gu  …  37.584848  127.094023

[10 rows x 5 columns]
    Station code Station name(district)  …   Latitude   Longitude
15           116             Gangseo-gu  …  37.544640  126.835151
16           117                Guro-gu  …  37.498498  126.889692
17           118           Geumcheon-gu  …  37.452357  126.908296
18           119        Yeongdeungpo-gu  …  37.525007  126.897370
19           120             Dongjak-gu  …  37.480917  126.971481
20           121              Gwanak-gu  …  37.487355  126.927102
21           122              Seocho-gu  …  37.504547  126.994458
22           123             Gangnam-gu  …  37.517528  127.047470
23           124              Songpa-gu  …  37.502686  127.092509
24           125            Gangdong-gu  …  37.544962  127.136792

[10 rows x 5 columns]
```

[64]:
```
# 06.02.09.07
# return dimensions
# dt03-dt05

print(dt03_measurement_info________00.shape)
print('-----------------------------------')
print(dt04_measurement_info_item___00.shape)
print('-----------------------------------')
print(dt05_measurement_info_stn____00.shape)
```

```
(3885066, 5)
-----------------------------------
(6, 7)
-----------------------------------
(25, 5)
```

[65]:
```
# 06.02.09.08
# confirm column names
# dt03-dt05

print(dt03_measurement_info________00.columns)
print('-----------------------------------')
print(dt04_measurement_info_item___00.columns)
print('-----------------------------------')
print(dt05_measurement_info_stn____00.columns)
```

```
Index(['Measurement date', 'Station code', 'Item code', 'Average value',
       'Instrument status'],
      dtype='object')
-----------------------------------
Index(['Item code', 'Item name', 'Unit of measurement', 'Good(Blue)',
       'Normal(Green)', 'Bad(Yellow)', 'Very bad(Red)'],
      dtype='object')
-----------------------------------
Index(['Station code', 'Station name(district)', 'Address', 'Latitude',
       'Longitude'],
      dtype='object')
```

[66]:
```
# 06.02.09.09
# column rename to remove spaces and quotes
# dt03

dt03_measurement_info________00_rn = dt03_measurement_info________00.
 ↪rename(columns = {
    'Measurement date': 'date_measurement',
    'Station code': 'code_station',
    'Item code': 'code_item',
    'Average value': 'value_ave',
```

```
    'Instrument status': 'status_instrument'
    })
```

[67]:
```
# 06.02.09.10
# column rename to remove spaces and quotes
# dt04

dt04_measurement_info_item___00_rn = dt04_measurement_info_item___00.
 ↪rename(columns = {
    'Item code': 'code_item',
    'Item name': 'name_item',
    'Unit of measurement': 'unit_measurement',
    'Good(Blue)': 'good_blue',
    'Normal(Green)': 'normal_green',
    'Bad(Yellow)': 'bad_yellow',
    'Very bad(Red)': 'very_bad_red'
    })
```

[68]:
```
# 06.02.09.11
# column rename to remove spaces and quotes
# dt05

dt05_measurement_info_stn____00_rn = dt05_measurement_info_stn____00.
 ↪rename(columns = {
    'Station code': 'code_station',
    'Station name(district)': 'name_station',
    'Address': 'address',
    'Latitude': 'lat',
    'Longitude': 'lon'
    })
```

[69]:
```
# 06.02.09.12
# confirm column names
# dt03-dt05

print(dt03_measurement_info________00_rn.columns)
print('----------------------------------------')
print(dt04_measurement_info_item___00_rn.columns)
print('----------------------------------------')
print(dt05_measurement_info_stn____00_rn.columns)
```

```
Index(['date_measurement', 'code_station', 'code_item', 'value_ave',
       'status_instrument'],
      dtype='object')
----------------------------------------
Index(['code_item', 'name_item', 'unit_measurement', 'good_blue',
       'normal_green', 'bad_yellow', 'very_bad_red'],
      dtype='object')
```

```
----------------------------------------
Index(['code_station', 'name_station', 'address', 'lat', 'lon'], dtype='object')
```

[70]:
```
# 06.02.10.01
# confirm types
# dt03

print(dt03_measurement_info________00_rn['date_measurement'].dtypes)
print(dt03_measurement_info________00_rn['code_station'].dtypes)
print(dt03_measurement_info________00_rn['code_item'].dtypes)
print(dt03_measurement_info________00_rn['value_ave'].dtypes)
print(dt03_measurement_info________00_rn['status_instrument'].dtypes)
```

```
object
int64
int64
float64
int64
```

[71]:
```
# 06.02.10.02
# confirm types
# dt04

print(dt04_measurement_info_item___00_rn['code_item'].dtypes)
print(dt04_measurement_info_item___00_rn['name_item'].dtypes)
print(dt04_measurement_info_item___00_rn['unit_measurement'].dtypes)
print(dt04_measurement_info_item___00_rn['good_blue'].dtypes)
print(dt04_measurement_info_item___00_rn['normal_green'].dtypes)
print(dt04_measurement_info_item___00_rn['bad_yellow'].dtypes)
print(dt04_measurement_info_item___00_rn['very_bad_red'].dtypes)
```

```
int64
object
object
float64
float64
float64
float64
```

[72]:
```
# 06.02.10.03
# confirm types
# dt05

print(dt05_measurement_info_stn____00_rn['code_station'].dtypes)
print(dt05_measurement_info_stn____00_rn['name_station'].dtypes)
print(dt05_measurement_info_stn____00_rn['address'].dtypes)
print(dt05_measurement_info_stn____00_rn['lat'].dtypes)
print(dt05_measurement_info_stn____00_rn['lon'].dtypes)
```

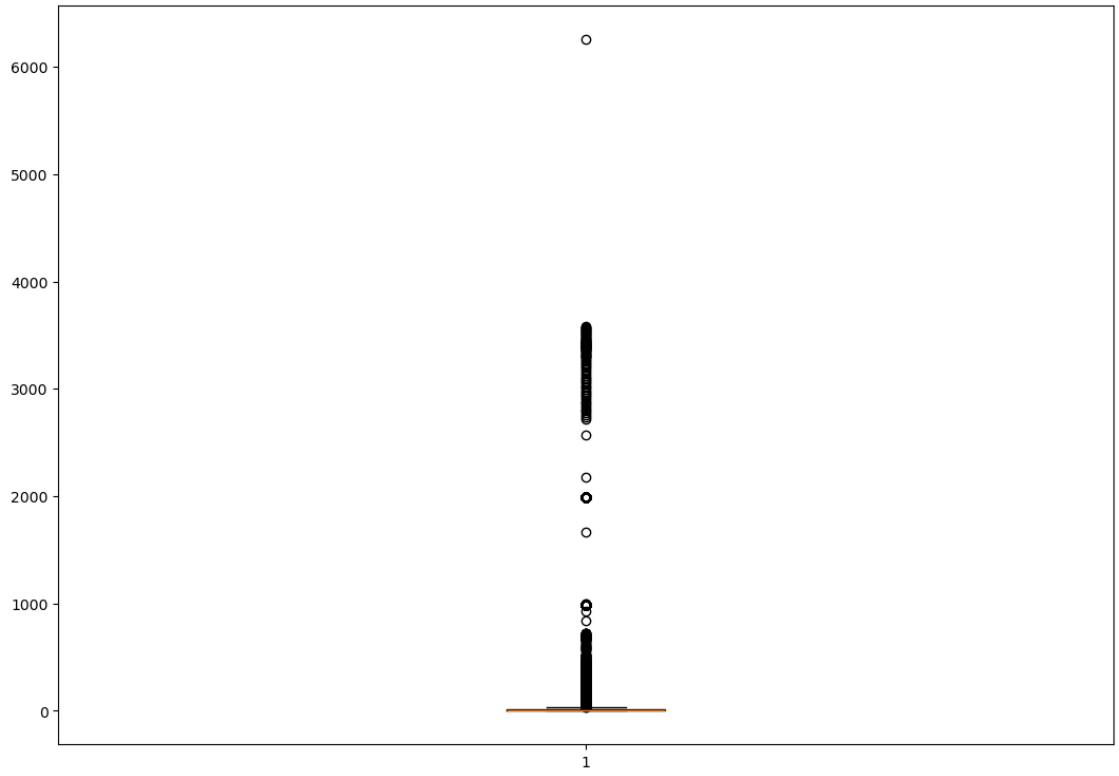
```
int64
object
object
float64
float64
```

[73]:
```
# 06.02.10.04
# change column to datetime
# dt03

ts, ms = '20.12.2016 09:38:42,76'.split(',')
dt03_measurement_info________00_rn['date_measure'] = datetime.strptime(ts, '%d.
 ↪%m.%Y %H:%M:%S')
dt03_measurement_info________00_rn_cv = dt03_measurement_info________00_rn
```

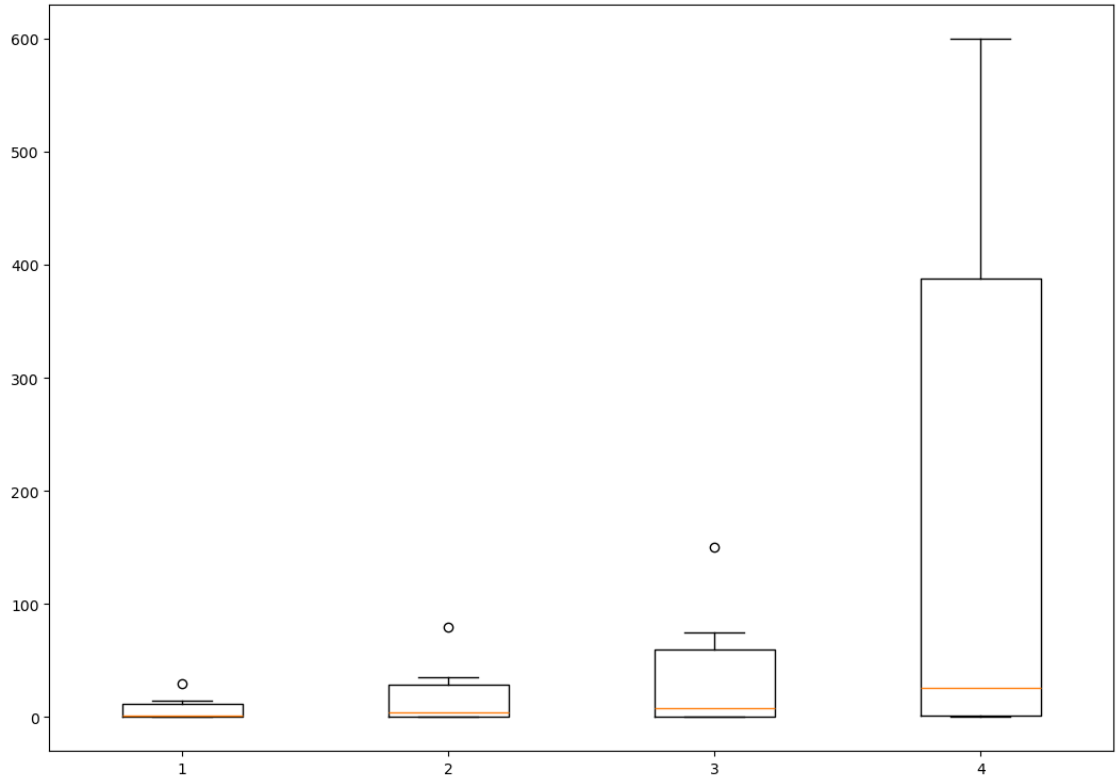
[74]:
```
# 06.02.10.05
# change column to float
# dt03

'''dt01_Air_Quality_Seoul_______00_rn_cv['SO2'] =␣
 ↪dt01_Air_Quality_Seoul_______00_rn['SO2'].astype(float)'''
```

[74]:
```
"dt01_Air_Quality_Seoul_______00_rn_cv['SO2'] =
dt01_Air_Quality_Seoul_______00_rn['SO2'].astype(float)"
```

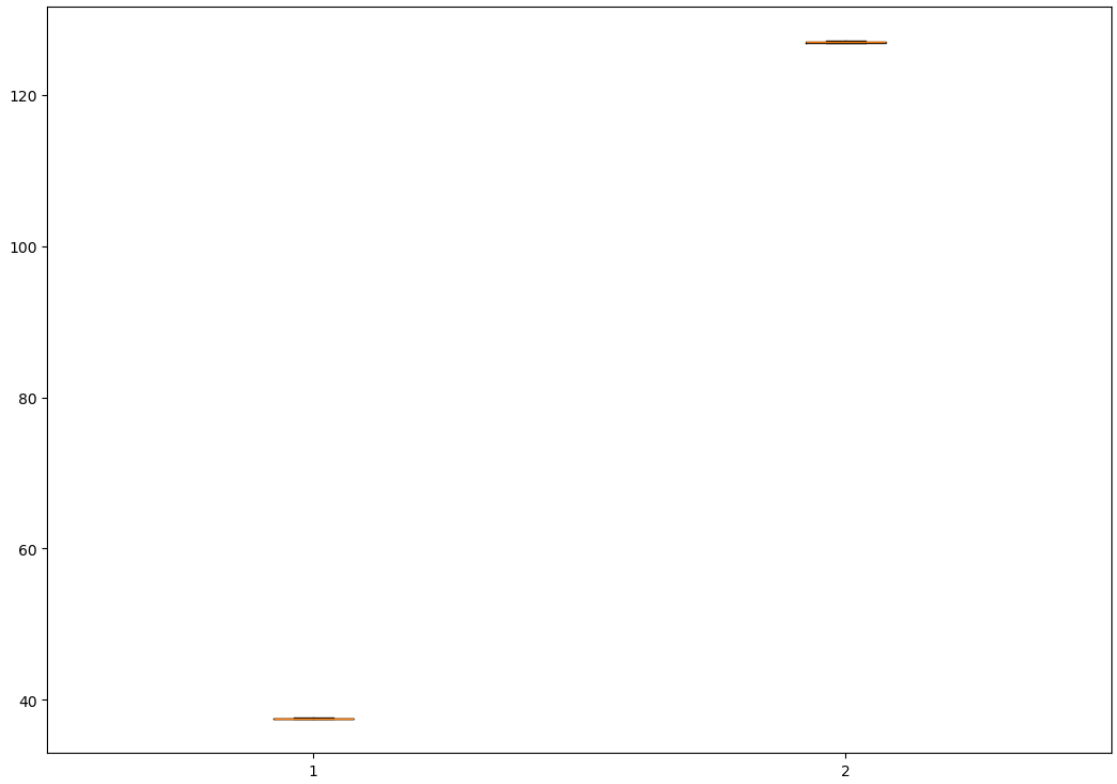
[75]:
```
# 06.02.11.01
# boxplot
# dt03

np.random.seed(10)
dt03_measurement_info________00_rn_boxplot =␣
 ↪[dt03_measurement_info________00_rn['value_ave']]
fig = plt.figure(figsize =(10, 7))
ax = fig.add_axes([0, 0, 1, 1])
bp = ax.boxplot(dt03_measurement_info________00_rn_boxplot)
plt.show()
```



[76]:
```
# 06.02.11.02
# boxplot
# dt04

np.random.seed(10)
dt04_measurement_info_item___00_rn_boxplot =␣
 ↪[dt04_measurement_info_item___00_rn['good_blue'],␣
 ↪dt04_measurement_info_item___00_rn['normal_green'],␣
 ↪dt04_measurement_info_item___00_rn['bad_yellow'],␣
 ↪dt04_measurement_info_item___00_rn['very_bad_red']]
fig = plt.figure(figsize =(10, 7))
ax = fig.add_axes([0, 0, 1, 1])
bp = ax.boxplot(dt04_measurement_info_item___00_rn_boxplot)
plt.show()
```



[77]:
```
# 06.02.11.03
# boxplot
# dt05

np.random.seed(10)
dt05_measurement_info_stn____00_rn_boxplot =␣
 ↪[dt05_measurement_info_stn____00_rn['lat'],␣
 ↪dt05_measurement_info_stn____00_rn['lon']]
fig = plt.figure(figsize =(10, 7))
ax = fig.add_axes([0, 0, 1, 1])
bp = ax.boxplot(dt05_measurement_info_stn____00_rn_boxplot)
plt.show()
```



[78]:
```
# 06.02.12.01
# preparing data for modeling
# add index column
# index column to select specific rows
# dt03

'''dt03_measurement_info________00_rn.insert(0, 'index', range(0, 0 +␣
↪len(dt03_measurement_info________00_rn)))'''
```

[78]:
```
"dt03_measurement_info________00_rn.insert(0, 'index', range(0, 0 +
len(dt03_measurement_info________00_rn)))"
```

[79]:
```
# 06.02.12.02
# preparing data for modeling
# sparse to partition data
# dt03

'''dt03_measurement_info________00_rn.to_sparse()'''
```

[79]:
```
'dt03_measurement_info________00_rn.to_sparse()'
```

[80]:
```
# 06.02.12.03
# preparing data for modeling
# create dummy variables
# due to returning boolean values, converting dummmies to integers
# dt03

'''dt03_measurement_info________00_rn_dv = pd.
 ↪get_dummies(dt03_measurement_info________00_rn, drop_first = True, dtype =␣
 ↪int)'''
```

[80]:
```
'dt03_measurement_info________00_rn_dv =
pd.get_dummies(dt03_measurement_info________00_rn, drop_first = True, dtype =
int)'
```

[81]:
```
# 06.02.12.04
# preparing data for modeling
# split data
# select columns
# dt03

'''dt03_x01 = dt03_measurement_info________00_rn_dv.drop(['value_ave'], axis =␣
 ↪1)
dt03_y01 = dt03_measurement_info________00_rn_dv['value_ave']'''
```

[81]:
```
"dt03_x01 = dt03_measurement_info________00_rn_dv.drop(['value_ave'], axis =
1)\ndt03_y01 = dt03_measurement_info________00_rn_dv['value_ave']"
```

[82]:
```
# 06.02.12.05
# preparing data for modeling
# split into train and test
# dt03

'''dt03_x01_trn, dt03_x01_tst, dt03_y01_trn, dt03_y01_tst =␣
 ↪train_test_split(dt03_x01, dt03_y01, test_size = 0.3, random_state = 0)'''
```

[82]:
```
'dt03_x01_trn, dt03_x01_tst, dt03_y01_trn, dt03_y01_tst =
train_test_split(dt03_x01, dt03_y01, test_size = 0.3, random_state = 0)'
```

[83]:
```
# 06.02.12.06
# preparing data for modeling
# assign regression variable
# dt03

'''dt03_lr01 = LinearRegression()'''
```

[83]:
```
'dt03_lr01 = LinearRegression()'
```

[84]:
```
# 06.02.12.07
# fit data for modeling
# fit variables to model
# dt03

'''dt03_lr01.fit(dt03_x01_trn, dt03_y01_trn)'''
```

[84]:
```
'dt03_lr01.fit(dt03_x01_trn, dt03_y01_trn)'
```

[85]:
```
# 06.02.12.08
# predict data for modeling
# fit variables to model
# dt03

'''dt03_y01_pdct = dt03_lr01.predict(dt03_x01_tst)'''
```

[85]:
```
'dt03_y01_pdct = dt03_lr01.predict(dt03_x01_tst)'
```

[86]:
```
# 06.02.12.09
# preparing data for modeling
# assign variable for rmse and r2
# dt03

'''dt03_rmse01 = np.sqrt(mean_squared_error(dt03_y01_tst, dt03_y01_pdct))
dt03_r201 = r2_score(dt03_y01_tst, dt03_y01_pdct)'''
```

[86]:
```
'dt03_rmse01 = np.sqrt(mean_squared_error(dt03_y01_tst,
dt03_y01_pdct))\ndt03_r201 = r2_score(dt03_y01_tst, dt03_y01_pdct)'
```

[87]:
```
# 06.02.12.10
# run model
# return rmse and r2 dt03
# rmse:
# r2:
# dt03

'''print(f'rmse: {dt03_rmse01}')
print(f'r2: {dt03_r201}')'''
```

[87]:
```
"print(f'rmse: {dt03_rmse01}')\nprint(f'r2: {dt03_r201}')"
```

[88]:
```
# 06.02.12.11
# assign variable for pca
# dt03

'''pca = PCA(.9)'''
```

[88]:
```
'pca = PCA(.9)'
```

[89]:
```
# 06.02.12.12
# calculate pca
# dt03

'''pca.fit(dt03_x01_trn)
dt03_x01_pca_trn = pca.transform(dt03_x01_trn)
dt03_x01_pca_tst = pca.transform(dt03_x01_tst)'''
```

[89]:
```
'pca.fit(dt03_x01_trn)\ndt03_x01_pca_trn =
pca.transform(dt03_x01_trn)\ndt03_x01_pca_tst = pca.transform(dt03_x01_tst)'
```

[90]:
```
# 06.02.12.13
# return pca calculation matrix
# dt03

'''print(f'features in pca matrix: {dt03_x01_pca_trn.shape[1]}')'''
```

[90]:
```
"print(f'features in pca matrix: {dt03_x01_pca_trn.shape[1]}')"
```

[91]:
```
# 06.02.13.01
# preparing data for modeling
# add index column
# index column to select specific rows
# dt04

dt04_measurement_info_item___00_rn.insert(0, 'index', range(0, 0 +␣
 ↪len(dt04_measurement_info_item___00_rn)))
```

[92]:
```
# 06.02.13.02
# preparing data for modeling
# create dummy variables
# due to returning boolean values, converting dummmies to integers
# dt04

dt04_measurement_info_item___00_rn_dv = pd.
 ↪get_dummies(dt04_measurement_info_item___00_rn, drop_first = True, dtype =␣
 ↪int)
```

[93]:
```
# 06.02.13.03
# preparing data for modeling
# split data
# select columns
# dt04

dt04_x01 = dt04_measurement_info_item___00_rn_dv.drop(['good_blue'], axis = 1)
dt04_y01 = dt04_measurement_info_item___00_rn_dv['good_blue']
```

[94]:
```
# 06.02.13.04
# preparing data for modeling
# split into train and test
# dt04

dt04_x01_trn, dt04_x01_tst, dt04_y01_trn, dt04_y01_tst =␣
 ↪train_test_split(dt04_x01, dt04_y01, test_size = 0.3, random_state = 0)
```

[95]:
```
# 06.02.13.05
# preparing data for modeling
# assign regression variable
# dt04

dt04_lr01 = LinearRegression()
```

[96]:
```
# 06.02.13.06
# fit data for modeling
# fit variables to model
# dt04

dt04_lr01.fit(dt04_x01_trn, dt04_y01_trn)
```

[96]:
```
LinearRegression()
```

[97]:
```
# 06.02.13.07
# predict data for modeling
# fit variables to model
# dt04

dt04_y01_pdct = dt04_lr01.predict(dt04_x01_tst)
```

[98]:
```
# 06.02.13.08
# preparing data for modeling
# assign variable for rmse and r2
# dt04

dt04_rmse01 = np.sqrt(mean_squared_error(dt04_y01_tst, dt04_y01_pdct))
dt04_r201 = r2_score(dt04_y01_tst, dt04_y01_pdct)
```

[99]:
```
# 06.02.13.09
# run model
# return rmse and r2 dt04
# rmse: 6.188212520525532
# r2: 0.09363374676475833
# dt04

print(f'rmse: {dt04_rmse01}')
```

```
print(f'r2: {dt04_r201}')
```

```
rmse: 6.188212520525532
r2: 0.09363374676475833
```

[100]:
```
# 06.02.13.10
# assign variable for pca
# dt04

pca = PCA(.9)
```

[101]:
```
# 06.02.13.11
# calculate pca
# dt04

pca.fit(dt04_x01_trn)
dt04_x01_pca_trn = pca.transform(dt04_x01_trn)
dt04_x01_pca_tst = pca.transform(dt04_x01_tst)
```

[102]:
```
# 06.02.13.12
# return pca calculation matrix
# dt04

print(f'features in pca matrix: {dt04_x01_pca_trn.shape[1]}')
```

```
features in pca matrix: 1
```

[103]:
```
# 06.02.14.01
# preparing data for modeling
# add index column
# index column to select specific rows
# dt04

'''dt04_measurement_info_item___01_rn.insert(0, 'index', range(0, 0 +␣
 ↪len(dt04_measurement_info_item___00_rn)))'''
```

[103]:
```
"dt04_measurement_info_item___01_rn.insert(0, 'index', range(0, 0 +
len(dt04_measurement_info_item___00_rn)))"
```

[104]:
```
# 06.02.14.02
# preparing data for modeling
# create dummy variables
# due to returning boolean values, converting dummmies to integers
# dt04

dt04_measurement_info_item___01_rn_dv = pd.
 ↪get_dummies(dt04_measurement_info_item___00_rn, drop_first = True, dtype =␣
 ↪int)
```

[105]:
```
# 06.02.14.03
# preparing data for modeling
# split data
# select columns
# dt04

dt04_x02 = dt04_measurement_info_item___01_rn_dv.drop(['normal_green'], axis =␣
 ↪1)
dt04_y02 = dt04_measurement_info_item___01_rn_dv['normal_green']
```

[106]:
```
# 06.02.14.04
# preparing data for modeling
# split into train and test
# dt04

dt04_x02_trn, dt04_x02_tst, dt04_y02_trn, dt04_y02_tst =␣
 ↪train_test_split(dt04_x02, dt04_y02, test_size = 0.3, random_state = 0)
```

[107]:
```
# 06.02.14.05
# preparing data for modeling
# assign regression variable
# dt04

dt04_lr02 = LinearRegression()
```

[108]:
```
# 06.02.14.06
# fit data for modeling
# fit variables to model
# dt04

dt04_lr02.fit(dt04_x02_trn, dt04_y02_trn)
```

[108]:
```
LinearRegression()
```

[109]:
```
# 06.02.14.07
# predict data for modeling
# fit variables to model
# dt04

dt04_y02_pdct = dt04_lr02.predict(dt04_x02_tst)
```

[110]:
```
# 06.02.14.08
# preparing data for modeling
# assign variable for rmse and r2
# dt04

dt04_rmse02 = np.sqrt(mean_squared_error(dt04_y02_tst, dt04_y02_pdct))
```

```
dt04_r202 = r2_score(dt04_y02_tst, dt04_y02_pdct)
```

[111]:
```
# 06.02.14.09
# run model
# return rmse and r2 dt04
# rmse: 20.483506216544086
# r2: -1.482686549841422
# dt04

print(f'rmse: {dt04_rmse02}')
print(f'r2: {dt04_r202}')
```

```
rmse: 20.483506216544086
r2: -1.482686549841422
```

[112]:
```
# 06.02.14.10
# assign variable for pca
# dt04

pca = PCA(.9)
```

[113]:
```
# 06.02.14.11
# calculate pca
# dt04

pca.fit(dt04_x02_trn)
dt04_x02_pca_trn = pca.transform(dt04_x02_trn)
dt04_x02_pca_tst = pca.transform(dt04_x02_tst)
```

[114]:
```
# 06.02.14.12
# return pca calculation matrix
# dt04

print(f'features in pca matrix: {dt04_x02_pca_trn.shape[1]}')
```

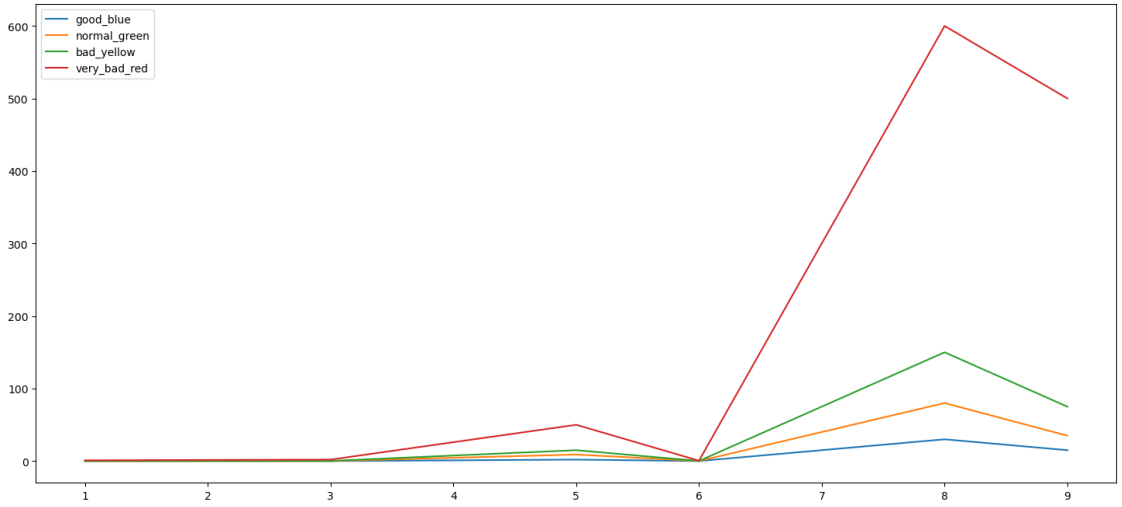
```
features in pca matrix: 1
```

[115]:
```
# 06.02.15.01
# render line plot
# dt03

'''dt03_plt_x00 = dt03_measurement_info________00_rn['date_measurement']
dt03_plt_y01 = dt03_measurement_info________00_rn['code_station']
dt03_plt_y02 = dt03_measurement_info________00_rn['code_item']
dt03_plt_y03 = dt03_measurement_info________00_rn['value_ave']
dt03_plt_y04 = dt03_measurement_info________00_rn['status_instrument']
plt.figure(figsize=(18,8))
plt.plot(dt03_plt_x00, dt03_plt_y01, label = "code_station", linestyle="-")
```

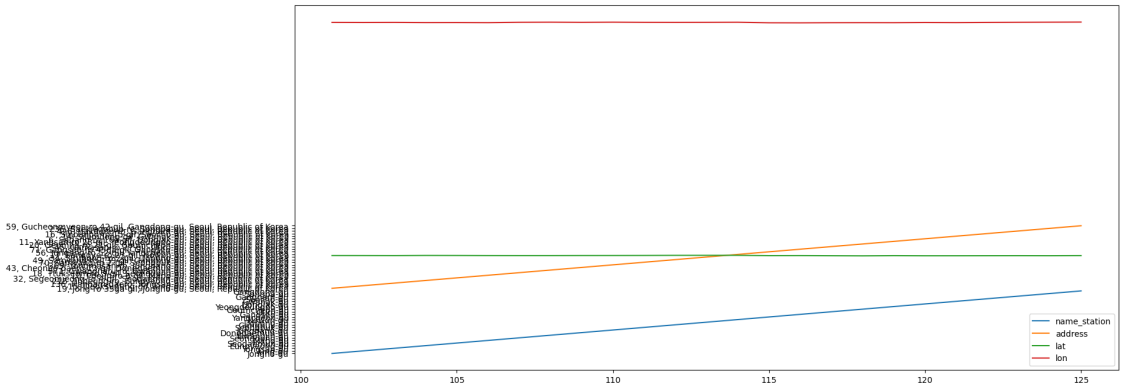
```
plt.plot(dt03_plt_x00, dt03_plt_y02, label = "code_item", linestyle="-")
plt.plot(dt03_plt_x00, dt03_plt_y03, label = "value_ave", linestyle="-")
plt.plot(dt03_plt_x00, dt03_plt_y04, label = "status_instrument", linestyle="-")
plt.legend()
plt.show()'''
```

[115]:
```
'dt03_plt_x00 =
dt03_measurement_info________00_rn[\'date_measurement\']\ndt03_plt_y01 =
dt03_measurement_info________00_rn[\'code_station\']\ndt03_plt_y02 =
dt03_measurement_info________00_rn[\'code_item\']\ndt03_plt_y03 =
dt03_measurement_info________00_rn[\'value_ave\']\ndt03_plt_y04 = dt03_measureme
nt_info________00_rn[\'status_instrument\']\nplt.figure(figsize=(18,8))\nplt.plo
t(dt03_plt_x00, dt03_plt_y01, label = "code_station",
linestyle="-")\nplt.plot(dt03_plt_x00, dt03_plt_y02, label = "code_item",
linestyle="-")\nplt.plot(dt03_plt_x00, dt03_plt_y03, label = "value_ave",
linestyle="-")\nplt.plot(dt03_plt_x00, dt03_plt_y04, label =
"status_instrument", linestyle="-")\nplt.legend() \nplt.show()'
```

[116]:
```
# 06.02.15.02
# render line plot
# dt04

dt04_plt_x00 = dt04_measurement_info_item___00_rn_dv['code_item']
dt04_plt_y01 = dt04_measurement_info_item___00_rn_dv['good_blue']
dt04_plt_y02 = dt04_measurement_info_item___00_rn_dv['normal_green']
dt04_plt_y03 = dt04_measurement_info_item___00_rn_dv['bad_yellow']
dt04_plt_y04 = dt04_measurement_info_item___00_rn_dv['very_bad_red']
plt.figure(figsize=(18,8))
plt.plot(dt04_plt_x00, dt04_plt_y01, label = "good_blue", linestyle="-")
plt.plot(dt04_plt_x00, dt04_plt_y02, label = "normal_green", linestyle="-")
plt.plot(dt04_plt_x00, dt04_plt_y03, label = "bad_yellow", linestyle="-")
plt.plot(dt04_plt_x00, dt04_plt_y04, label = "very_bad_red", linestyle="-")
plt.legend()
plt.show()
```



[117]:
```
# 06.02.15.03
# render line plot
# dt05

dt05_plt_x00 = dt05_measurement_info_stn____00_rn['code_station']
dt05_plt_y01 = dt05_measurement_info_stn____00_rn['name_station']
dt05_plt_y02 = dt05_measurement_info_stn____00_rn['address']
dt05_plt_y03 = dt05_measurement_info_stn____00_rn['lat']
dt05_plt_y04 = dt05_measurement_info_stn____00_rn['lon']
plt.figure(figsize=(18,8))
plt.plot(dt05_plt_x00, dt05_plt_y01, label = "name_station", linestyle="-")
plt.plot(dt05_plt_x00, dt05_plt_y02, label = "address", linestyle="-")
plt.plot(dt05_plt_x00, dt05_plt_y03, label = "lat", linestyle="-")
plt.plot(dt05_plt_x00, dt05_plt_y04, label = "lon", linestyle="-")
plt.legend()
plt.show()
```

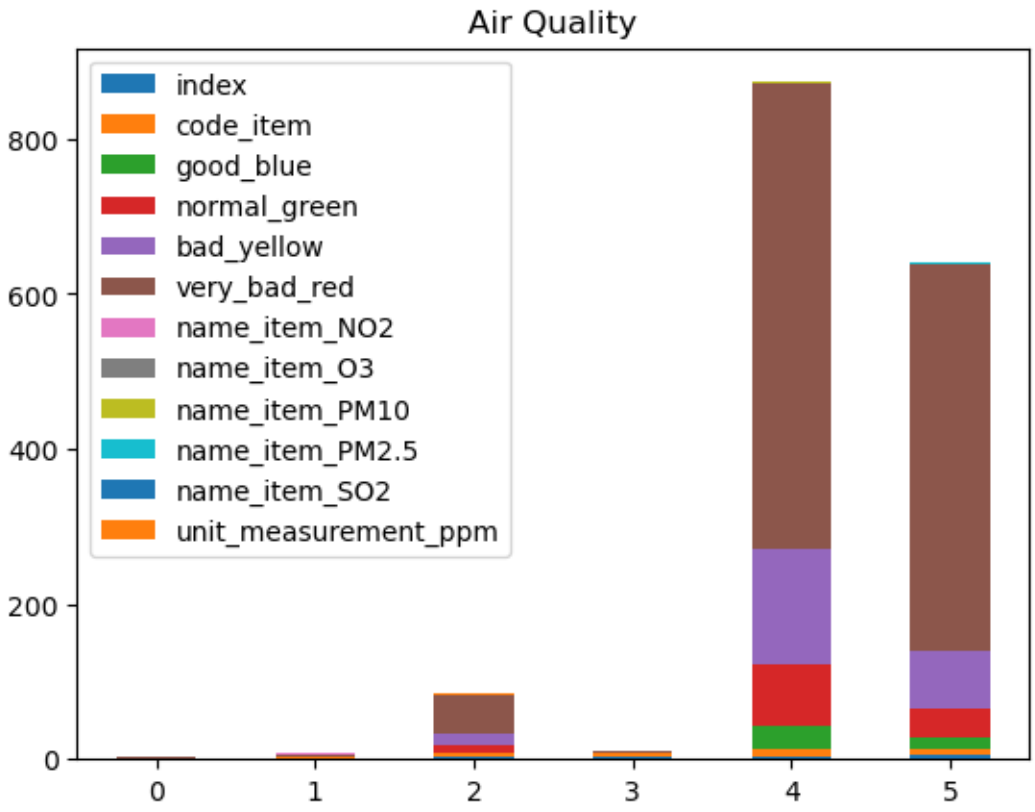


[118]:
```
# 06.02.15.04
# render bar chart
# dt03

'''plt.figure(figsize=(18,8))
dt03_measurement_info________00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()'''
```

[118]:
```
"plt.figure(figsize=(18,8))\ndt03_measurement_info________00_rn.plot(kind='bar',
stacked=True)\nplt.title('Air Quality')\nplt.xticks(rotation=0,
ha='center')\nplt.show()"
```

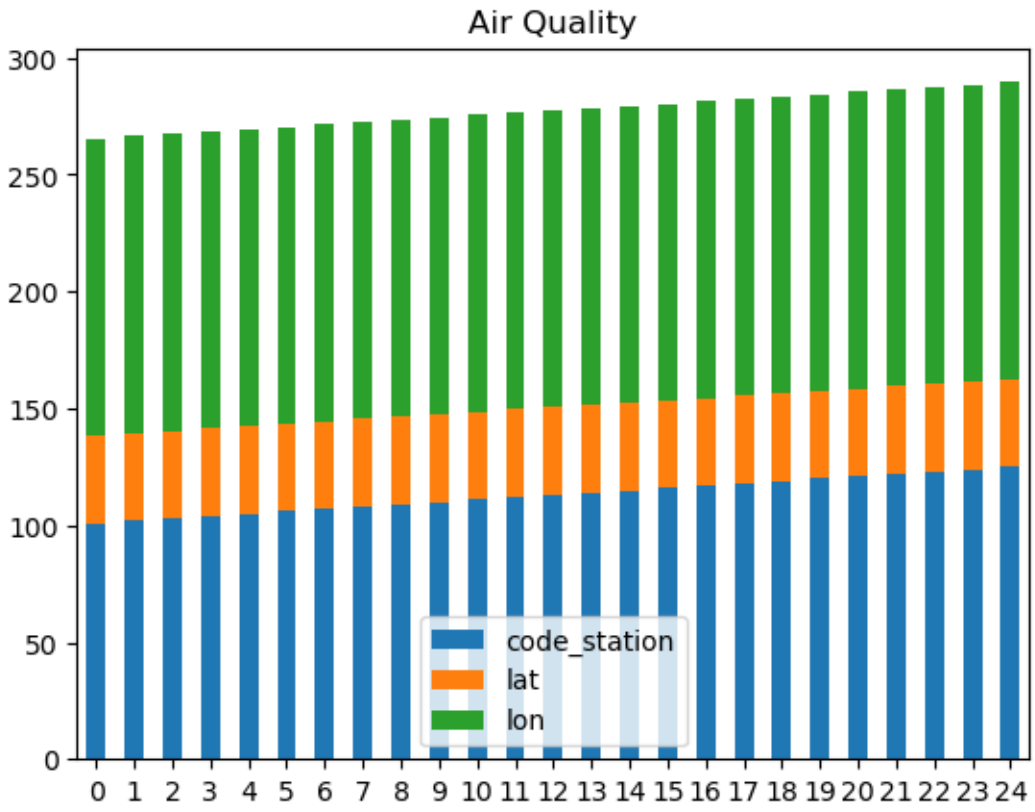
[119]:
```
# 06.02.15.05
# render bar chart
# dt04

plt.figure(figsize=(18,8))
dt04_measurement_info_item___00_rn_dv.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```



[120]:
```
# 06.02.15.06
# render bar chart
# dt05

plt.figure(figsize=(18,8))
dt05_measurement_info_stn____00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```



[121]:
```
# 06.02.16.01
# preparing data for modeling
# add index column
# index column to select specific rows
# dt05

dt05_measurement_info_stn____00_rn.insert(0, 'index', range(0, 0 +␣
 ↪len(dt05_measurement_info_stn____00_rn)))
```

[122]:
```
# 06.02.16.02
# preparing data for modeling
# create dummy variables
# due to returning boolean values, converting dummmies to integers
# dt05

dt05_measurement_info_stn____00_rn_dv = pd.
 ↪get_dummies(dt05_measurement_info_stn____00_rn, drop_first = True, dtype =␣
 ↪int)
```

[123]:
```
# 06.02.16.03
# preparing data for modeling
```

```
# split data
# select columns
# dt05

dt05_x01 = dt05_measurement_info_stn____00_rn_dv.drop(['lat'], axis = 1)
dt05_y01 = dt05_measurement_info_stn____00_rn_dv['lat']
```

[124]:
```
# 06.02.16.04
# preparing data for modeling
# split into train and test
# dt05

dt05_x01_trn, dt05_x01_tst, dt05_y01_trn, dt05_y01_tst =␣
↪train_test_split(dt05_x01, dt05_y01, test_size = 0.3, random_state = 0)
```

[125]:
```
# 06.02.16.05
# preparing data for modeling
# assign regression variable
# dt05

dt05_lr01 = LinearRegression()
```

[126]:
```
# 06.02.16.06
# fit data for modeling
# fit variables to model
# dt05

dt05_lr01.fit(dt05_x01_trn, dt05_y01_trn)
```

[126]:
```
LinearRegression()
```

[127]:
```
# 06.02.16.07
# predict data for modeling
# fit variables to model
# dt05

dt05_y01_pdct = dt05_lr01.predict(dt05_x01_tst)
```

[128]:
```
# 06.02.16.08
# preparing data for modeling
# assign variable for rmse and r2
# dt05

dt05_rmse01 = np.sqrt(mean_squared_error(dt05_y01_tst, dt05_y01_pdct))
dt05_r201 = r2_score(dt05_y01_tst, dt05_y01_pdct)
```

[129]:
```
# 06.02.16.09
# run model
# return rmse and r2 dt05
# rmse: 0.06575946272795244
# r2: -0.4136758087823822
# dt05

print(f'rmse: {dt05_rmse01}')
print(f'r2: {dt05_r201}')
```

```
rmse: 0.06557177964531924
r2: -0.40561782304382965
```

[130]:
```
# 06.02.16.10
# assign variable for pca
# dt05

pca = PCA(.9)
```

[131]:
```
# 06.02.16.11
# calculate pca
# dt05

pca.fit(dt05_x01_trn)
dt05_x01_pca_trn = pca.transform(dt05_x01_trn)
dt05_x01_pca_tst = pca.transform(dt05_x01_tst)
```

[132]:
```
# 06.02.16.12
# return pca calculation matrix
# dt05

print(f'features in pca matrix: {dt05_x01_pca_trn.shape[1]}')
```

```
features in pca matrix: 1
```

3 milestone 03

[133]:
```
# 06.02.17.01
# read csv
# assign variable
# dt06

dt06_dt01____________________00 = pd.
 ↪read_csv('106_DT_106N_03_0200045_20240703151242.csv')
```

[134]:
```
# 06.02.17.02
# read csv
# assign variable
```

```
# dt07

dt07_dt02_____________________00 = pd.
 ↪read_csv('106_DT_106N_03_0200076_20240703151123.csv')
```

[135]:
```
# 06.02.17.03
# read csv
# assign variable
# dt08

dt08_summary_measurement_____00 = pd.read_csv('Measurement_summary.csv')
```

[136]:
```
# 06.02.17.04
# read csv
# assign variable
# dt09

dt09_nat_emissions___________00 = pd.
 ↪read_csv('National_Air_Pollutant_Emissions_20240703151513.csv')
```

[137]:
```
# 06.02.17.05
# read csv
# assign variable
# dt10

dt10_seoul_air_____________00 = pd.read_csv('seoul_air_1988_2021.csv')
```

[138]:
```
# 06.02.17.06
# read csv
# assign variable
# dt11

dt11_seoul_ave_air__________00 = pd.read_csv('SeoulHourlyAvgAirPollution.csv')
```

[139]:
```
# 06.02.18.01
# return first and last ten rows
# dt06

print(dt06_dt01_____________________00.head(10))
print(dt06_dt01_____________________00.tail(10))
```

```
  [13101128237A]Classification Classification  … 2023.12 Month Unnamed: 173
0          13102128237A.4100001          Total  …            35          NaN
1          13102128237A.4200003          Seoul  …            36          NaN
2          13102128237A.4200005          Busan  …            31          NaN
3          13102128237A.4200007          Daegu  …            37          NaN
4          13102128237A.4200009        Incheon  …            39          NaN
5          13102128237A.4200011        Gwangju  …            32          NaN
```

```
6         13102128237A.4200013        Daejeon  …            34          NaN
7         13102128237A.4200015          Ulsan  …            32          NaN
8         13102128237A.4200017      Sejong-si  …            39          NaN
9         13102128237A.4200019          Suwon  …            37          NaN

[10 rows x 174 columns]
    [13101128237A]Classification  … Unnamed: 173
165         13102128237A.4200183  …          NaN
166         13102128237A.4200184  …          NaN
167         13102128237A.4200185  …          NaN
168         13102128237A.4200186  …          NaN
169         13102128237A.4200187  …          NaN
170         13102128237A.4200188  …          NaN
171         13102128237A.4200179  …          NaN
172         13102128237A.4200190  …          NaN
173         13102128237A.4200191  …          NaN
174         13102128237A.4200192  …          NaN

[10 rows x 174 columns]
```

[140]:
```
# 06.02.18.02
# return first and last ten rows
# dt07

print(dt07_dt02_____________________00.head(10))
print(dt07_dt02_____________________00.tail(10))
```

```
  Classification              Item  …  2023.12 Month  Unnamed: 171
0  Seosomun-dong  Average for month  …            NaN           NaN
1        Jung-gu  Average for month  …             32           NaN
2     Hyoje-dong  Average for month  …            NaN           NaN
3      Jongno-gu  Average for month  …             37           NaN
4  Myeonmok-dong  Average for month  …            NaN           NaN
5    Jungnang-gu  Average for month  …             30           NaN
6    Yongdu-dong  Average for month  …            NaN           NaN
7  Dongdaemun-gu  Average for month  …             37           NaN
8  Bulgwang-dong  Average for month  …            NaN           NaN
9   Eunpyeong-gu  Average for month  …             34           NaN

[10 rows x 172 columns]
           Classification              Item  …  2023.12 Month  Unnamed: 171
717           Geumam-dong  Average for month  …            NaN           NaN
718           Seosin-dong  Average for month  …             38           NaN
719          Geumma-myeon  Average for month  …             37           NaN
720  Average for provinces  Average for month  …             37           NaN
721           Uhyeon-dong  Average for month  …             34           NaN
722             Samjin-ro  Average for month  …             28           NaN
723             Bansongro  Average for month  …             29           NaN
```

```
724           Gimhae-daero  Average for month  …             31           NaN
725  Average for provinces  Average for month  …             29           NaN
726            Nohyeong-ro  Average for month  …             33           NaN

[10 rows x 172 columns]
```

[141]:
```
# 06.02.18.03
# return first and last ten rows
# dt08

print(dt08_summary_measurement_____00.head(10))
print(dt08_summary_measurement_____00.tail(10))
```

```
   Measurement date  Station code  …  PM10  PM2.5
0  2017-01-01 00:00           101  …  73.0   57.0
1  2017-01-01 01:00           101  …  71.0   59.0
2  2017-01-01 02:00           101  …  70.0   59.0
3  2017-01-01 03:00           101  …  70.0   58.0
4  2017-01-01 04:00           101  …  69.0   61.0
5  2017-01-01 05:00           101  …  70.0   61.0
6  2017-01-01 06:00           101  …  66.0   57.0
7  2017-01-01 07:00           101  …  71.0   60.0
8  2017-01-01 08:00           101  …  72.0   60.0
9  2017-01-01 09:00           101  …  74.0   63.0

[10 rows x 11 columns]
        Measurement date  Station code  …  PM10  PM2.5
647501  2019-12-31 14:00           125  …  23.0   18.0
647502  2019-12-31 15:00           125  …  24.0   18.0
647503  2019-12-31 16:00           125  …  27.0   18.0
647504  2019-12-31 17:00           125  …  27.0   19.0
647505  2019-12-31 18:00           125  …  24.0   18.0
647506  2019-12-31 19:00           125  …  23.0   17.0
647507  2019-12-31 20:00           125  …  25.0   19.0
647508  2019-12-31 21:00           125  …  24.0   17.0
647509  2019-12-31 22:00           125  …  25.0   18.0
647510  2019-12-31 23:00           125  …  27.0   18.0

[10 rows x 11 columns]
```

[142]:
```
# 06.02.18.04
# return first and last ten rows
# dt09

print(dt09_nat_emissions__________00.head(10))
print(dt09_nat_emissions__________00.tail(10))
```

```
                Division(1)  …          2021.8
0                Division(1)  …  Ammonia (NH©ý)
```

```
1          The entire nation  …       262008256
2         Seoul Special City  …         2951043
3    Busan Metropolitan City  …         1545069
4    Daegu Metropolitan City  …         1422113
5  Incheon Metropolitan City  …         5514503
6  Gwangju Metropolitan City  …          783581
7  Daejeon Metropolitan City  …          666444
8    Ulsan Metropolitan City  …        12655548
9                Sejong City  …         2493439

[10 rows x 181 columns]
                            Division(1)       1999  …     2021.7    2021.8
10                    Gyeongggi Province  153664354  …  188788604  36233165
11                      Gangwon Province   34271959  …   26260521  13354923
12           North Chungcheong Province   37881702  …   39758294  13693366
13           South Chungcheong Province   44431869  …   78180901  43536060
14               North Jeonlla Province   39450484  …   67464898  26321081
15               South Jeonlla Province   47668516  …   99217389  38309256
16            North Gyeongsang Province   64463678  …   83054466  32619409
17            South Gyeongsang Province   53770091  …   94574671  23567347
18  Jeju Special Self-Governing Province   10971963  …   15353547   6333985
19                                 ocean          -  …   15420800      7923

[10 rows x 181 columns]
```

[143]:
```
# 06.02.18.05
# return first and last ten rows
# dt10

print(dt10_seoul_air_____________00.head(10))
print(dt10_seoul_air_____________00.tail(10))
```

```
           dt  loc        lat        long  …    co     o3  pm10  pm2.5
0  1988010100  103  37.540037  127.002661  …  10.3  0.000   NaN    NaN
1  1988010100  105  37.593730  126.947561  …  12.6  0.043   NaN    NaN
2  1988010100  107  37.542043  127.047497  …  13.4    NaN   NaN    NaN
3  1988010100  108  37.547185  127.090304  …   5.4  0.000   NaN    NaN
4  1988010100  113  37.654140  127.026801  …  14.6  0.000   NaN    NaN
5  1988010100  117  37.498268  126.887930  …  15.6  0.000   NaN    NaN
6  1988010100  122  37.504547  126.992308  …  12.7  0.000   NaN    NaN
7  1988010100  124  37.502688  127.090327  …   7.1  0.000   NaN    NaN
8  1988010101  103  37.540037  127.002661  …  13.7  0.000   NaN    NaN
9  1988010101  105  37.593730  126.947561  …  18.9  0.044   NaN    NaN

[10 rows x 10 columns]
                 dt  loc        lat        long  …   co     o3  pm10  pm2.5
5984772  2021123123  116  37.544644  126.832962  …  0.4  0.026  22.0    7.0
5984773  2021123123  117  37.498268  126.887930  …  0.3  0.024  21.0    4.0
```

```
5984774  2021123123  118  37.452357  126.906096  …  0.4  0.019  19.0    7.0
5984775  2021123123  119  37.526348  126.894067  …  0.4  0.019  22.0    9.0
5984776  2021123123  120  37.480932  126.969409  …  0.4  0.022  24.0   11.0
5984777  2021123123  121  37.487359  126.924913  …  0.5  0.016  21.0    9.0
5984778  2021123123  122  37.504547  126.992308  …  0.4  0.023  16.0   10.0
5984779  2021123123  123  37.517546  127.045775  …  0.4  0.024  18.0    6.0
5984780  2021123123  124  37.502688  127.090327  …  0.5  0.020  21.0    8.0
5984781  2021123123  125  37.544989  127.134599  …  0.4  0.020  22.0   10.0

[10 rows x 10 columns]
```

[144]:
```
# 06.02.18.06
# return first and last ten rows
# dt11

print(dt11_seoul_ave_air__________00.head(10))
print(dt11_seoul_ave_air__________00.tail(10))
```

```
                    (ppm)  …     (ppm)    ( / )     ( / )
0  201711242300          0.038  …       0.005       16.0        10.0
1  201711242200          0.031  …       0.005       17.0         9.0
2  201711242100          0.025  …       0.005       18.0        11.0
3  201711242000          0.033  …       0.005       21.0        12.0
4  201711241900          0.033  …       0.005       20.0        10.0
5  201711241800          0.026  …       0.005       21.0        10.0
6  201711241700          0.021  …       0.005       21.0        13.0
7  201711241600          0.017  …       0.005       19.0        11.0
8  201711241500          0.015  …       0.004       21.0        10.0
9  201711241400          0.015  …       0.005       21.0        10.0

[10 rows x 8 columns]
                       (ppm)  …     (ppm)    ( / )     ( / )
4215  201711180800          0.019  …       0.007       30.0        23.0
4216  201711180700          0.015  …       0.005       38.0        30.0
4217  201711180600          0.014  …       0.006       50.0        46.0
4218  201711180500          0.013  …       0.006       67.0        55.0
4219  201711180400          0.014  …       0.006       60.0        54.0
4220  201711180300          0.013  …       0.006       46.0        38.0
4221  201711180200          0.014  …       0.006       34.0        25.0
4222  201711180100          0.019  …       0.006       36.0        22.0
4223  201711180000          0.017  …       0.006       35.0        28.0
4224  201711172300          0.038  …       0.008       50.0        34.0

[10 rows x 8 columns]
```

[145]:
```
# 06.02.19.01
# return dimensions
# dt06-dt11
```

```
print(dt06_dt01_____________________00.shape)
print('-----------------------------------')
print(dt07_dt02_____________________00.shape)
print('-----------------------------------')
print(dt08_summary_measurement_____00.shape)
print('-----------------------------------')
print(dt09_nat_emissions__________00.shape)
print('-----------------------------------')
print(dt10_seoul_air_____________00.shape)
print('-----------------------------------')
print(dt11_seoul_ave_air__________00.shape)
```

```
(175, 174)
-----------------------------------
(727, 172)
-----------------------------------
(647511, 11)
-----------------------------------
(20, 181)
-----------------------------------
(5984782, 10)
-----------------------------------
(4225, 8)
```

[146]:
```
# 06.02.19.02
# confirm column names
# dt06-dt11

print(dt06_dt01_____________________00.columns)
print('-----------------------------------')
print(dt07_dt02_____________________00.columns)
print('-----------------------------------')
print(dt08_summary_measurement_____00.columns)
print('-----------------------------------')
print(dt09_nat_emissions__________00.columns)
print('-----------------------------------')
print(dt08_summary_measurement_____00.columns)
print('-----------------------------------')
print(dt10_seoul_air_____________00.columns)
print('-----------------------------------')
print(dt11_seoul_ave_air__________00.columns)
```

```
Index(['[13101128237A]Classification', 'Classification', '[Item]Item', 'Item',
       'UNIT', '2010.01 Month', '2010.02 Month', '2010.03 Month',
       '2010.04 Month', '2010.05 Month',
       …
       '2023.04 Month', '2023.05 Month', '2023.06 Month', '2023.07 Month',
```

```
       '2023.08 Month', '2023.09 Month', '2023.10 Month', '2023.11 Month',
       '2023.12 Month', 'Unnamed: 173'],
      dtype='object', length=174)
-------------------------------------
Index(['Classification', 'Item', 'UNIT', '2010.01 Month', '2010.02 Month',
       '2010.03 Month', '2010.04 Month', '2010.05 Month', '2010.06 Month',
       '2010.07 Month',
       …
       '2023.04 Month', '2023.05 Month', '2023.06 Month', '2023.07 Month',
       '2023.08 Month', '2023.09 Month', '2023.10 Month', '2023.11 Month',
       '2023.12 Month', 'Unnamed: 171'],
      dtype='object', length=172)
-------------------------------------
Index(['Measurement date', 'Station code', 'Address', 'Latitude', 'Longitude',
       'SO2', 'NO2', 'O3', 'CO', 'PM10', 'PM2.5'],
      dtype='object')
-------------------------------------
Index(['Division(1)', '1999', '1999.1', '1999.2', '1999.3', '1999.4', '1999.5',
       '1999.6', '2000', '2000.1',
       …
       '2020.8', '2021', '2021.1', '2021.2', '2021.3', '2021.4', '2021.5',
       '2021.6', '2021.7', '2021.8'],
      dtype='object', length=181)
-------------------------------------
Index(['Measurement date', 'Station code', 'Address', 'Latitude', 'Longitude',
       'SO2', 'NO2', 'O3', 'CO', 'PM10', 'PM2.5'],
      dtype='object')
-------------------------------------
Index(['dt', 'loc', 'lat', 'long', 'so2', 'no2', 'co', 'o3', 'pm10', 'pm2.5'],
dtype='object')
-------------------------------------
Index(['  ', '  ', '    (ppm)', '  (ppm)', '    (ppm)',
       '   (ppm)', '  ( / )', '   ( / )'],
      dtype='object')
```

[147]:
```
# 06.02.20.01
# column rename to remove spaces and quotes
# dt06

dt06_dt01_____________________00_rn = dt06_dt01_____________________00.
 ↪rename(columns = {
    '[13101128237A]Classification': 'classification_01',
    'Classification': 'classification_02',
    '[Item]Item': 'item_01',
    'Item': 'item_02',
    'UNIT': 'unit',
    '2010.01 Month': '2010_01',
```

```
    '2010.02 Month': '2010_02',
    '2010.03 Month': '2010_03',
    '2010.04 Month': '2010_04',
    '2010.05 Month': '2010_05',
    '2010.06 Month': '2010_06',
    '2010.07 Month': '2010_07',
    '2010.08 Month': '2010_08',
    '2010.09 Month': '2010_09',
    '2010.10 Month': '2010_10',
    '2010.11 Month': '2010_11',
    '2010.12 Month': '2010_12',
    '2011.01 Month': '2011_01',
    '2011.02 Month': '2011_02',
    '2011.03 Month': '2011_03',
    '2011.04 Month': '2011_04',
    '2011.05 Month': '2011_05',
    '2011.06 Month': '2011_06',
    '2011.07 Month': '2011_07',
    '2011.08 Month': '2011_08',
    '2011.09 Month': '2011_09',
    '2011.10 Month': '2011_10',
    '2011.11 Month': '2011_11',
    '2011.12 Month': '2011_12',
    '2012.01 Month': '2012_01',
    '2012.02 Month': '2012_02',
    '2012.03 Month': '2012_03',
    '2012.04 Month': '2012_04',
    '2012.05 Month': '2012_05',
    '2012.06 Month': '2012_06',
    '2012.07 Month': '2012_07',
    '2012.08 Month': '2012_08',
    '2012.09 Month': '2012_09',
    '2012.10 Month': '2012_10',
    '2012.11 Month': '2012_11',
    '2012.12 Month': '2012_12',
    '2013.01 Month': '2013_01',
    '2013.02 Month': '2013_02',
    '2013.03 Month': '2013_03',
    '2013.04 Month': '2013_04',
    '2013.05 Month': '2013_05',
    '2013.06 Month': '2013_06',
    '2013.07 Month': '2013_07',
    '2013.08 Month': '2013_08',
    '2013.09 Month': '2013_09',
    '2013.10 Month': '2013_10',
    '2013.11 Month': '2013_11',
    '2013.12 Month': '2013_12',
```

```
    '2014.01 Month': '2014_01',
    '2014.02 Month': '2014_02',
    '2014.03 Month': '2014_03',
    '2014.04 Month': '2014_04',
    '2014.05 Month': '2014_05',
    '2014.06 Month': '2014_06',
    '2014.07 Month': '2014_07',
    '2014.08 Month': '2014_08',
    '2014.09 Month': '2014_09',
    '2014.10 Month': '2014_10',
    '2014.11 Month': '2014_11',
    '2014.12 Month': '2014_12',
    '2015.01 Month': '2015_01',
    '2015.02 Month': '2015_02',
    '2015.03 Month': '2015_03',
    '2015.04 Month': '2015_04',
    '2015.05 Month': '2015_05',
    '2015.06 Month': '2015_06',
    '2015.07 Month': '2015_07',
    '2015.08 Month': '2015_08',
    '2015.09 Month': '2015_09',
    '2015.10 Month': '2015_10',
    '2015.11 Month': '2015_11',
    '2015.12 Month': '2015_12',
    '2016.01 Month': '2016_01',
    '2016.02 Month': '2016_02',
    '2016.03 Month': '2016_03',
    '2016.04 Month': '2016_04',
    '2016.05 Month': '2016_05',
    '2016.06 Month': '2016_06',
    '2016.07 Month': '2016_07',
    '2016.08 Month': '2016_08',
    '2016.09 Month': '2016_09',
    '2016.10 Month': '2016_10',
    '2016.11 Month': '2016_11',
    '2016.12 Month': '2016_12',
    '2017.01 Month': '2017_01',
    '2017.02 Month': '2017_02',
    '2017.03 Month': '2017_03',
    '2017.04 Month': '2017_04',
    '2017.05 Month': '2017_05',
    '2017.06 Month': '2017_06',
    '2017.07 Month': '2017_07',
    '2017.08 Month': '2017_08',
    '2017.09 Month': '2017_09',
    '2017.10 Month': '2017_10',
    '2017.11 Month': '2017_11',
```

```
    '2017.12 Month': '2017_12',
    '2018.01 Month': '2018_01',
    '2018.02 Month': '2018_02',
    '2018.03 Month': '2018_03',
    '2018.04 Month': '2018_04',
    '2018.05 Month': '2018_05',
    '2018.06 Month': '2018_06',
    '2018.07 Month': '2018_07',
    '2018.08 Month': '2018_08',
    '2018.09 Month': '2018_09',
    '2018.10 Month': '2018_10',
    '2018.11 Month': '2018_11',
    '2018.12 Month': '2018_12',
    '2019.01 Month': '2019_01',
    '2019.02 Month': '2019_02',
    '2019.03 Month': '2019_03',
    '2019.04 Month': '2019_04',
    '2019.05 Month': '2019_05',
    '2019.06 Month': '2019_06',
    '2019.07 Month': '2019_07',
    '2019.08 Month': '2019_08',
    '2019.09 Month': '2019_09',
    '2019.10 Month': '2019_10',
    '2019.11 Month': '2019_11',
    '2019.12 Month': '2019_12',
    '2020.01 Month': '2020_01',
    '2020.02 Month': '2020_02',
    '2020.03 Month': '2020_03',
    '2020.04 Month': '2020_04',
    '2020.05 Month': '2020_05',
    '2020.06 Month': '2020_06',
    '2020.07 Month': '2020_07',
    '2020.08 Month': '2020_08',
    '2020.09 Month': '2020_09',
    '2020.10 Month': '2020_10',
    '2020.11 Month': '2020_11',
    '2020.12 Month': '2020_12',
    '2021.01 Month': '2021_01',
    '2021.02 Month': '2021_02',
    '2021.03 Month': '2021_03',
    '2021.04 Month': '2021_04',
    '2021.05 Month': '2021_05',
    '2021.06 Month': '2021_06',
    '2021.07 Month': '2021_07',
    '2021.08 Month': '2021_08',
    '2021.09 Month': '2021_09',
    '2021.10 Month': '2021_10',
```

```
    '2021.11 Month': '2021_11',
    '2021.12 Month': '2021_12',
    '2022.01 Month': '2022_01',
    '2022.02 Month': '2022_02',
    '2022.03 Month': '2022_03',
    '2022.04 Month': '2022_04',
    '2022.05 Month': '2022_05',
    '2022.06 Month': '2022_06',
    '2022.07 Month': '2022_07',
    '2022.08 Month': '2022_08',
    '2022.09 Month': '2022_09',
    '2022.10 Month': '2022_10',
    '2022.11 Month': '2022_11',
    '2022.12 Month': '2022_12',
    '2023.01 Month': '2023_01',
    '2023.02 Month': '2023_02',
    '2023.03 Month': '2023_03',
    '2023.04 Month': '2023_04',
    '2023.05 Month': '2023_05',
    '2023.06 Month': '2023_06',
    '2023.07 Month': '2023_07',
    '2023.08 Month': '2023_08',
    '2023.09 Month': '2023_09',
    '2023.10 Month': '2023_10',
    '2023.11 Month': '2023_11',
    '2023.12 Month': '2023_12',
    'Unnamed: 173': 'status_instrument'
    })
```

[148]:
```
# 06.02.20.02
# column rename to remove spaces and quotes
# dt07

dt07_dt02_____________________00_rn = dt07_dt02_____________________00.
 ↪rename(columns = {
    'Classification': 'classification',
    'Item': 'item',
    'UNIT': 'unit',
    '2010.01 Month': '2010_01',
    '2010.02 Month': '2010_02',
    '2010.03 Month': '2010_03',
    '2010.04 Month': '2010_04',
    '2010.05 Month': '2010_05',
    '2010.06 Month': '2010_06',
    '2010.07 Month': '2010_07',
    '2010.08 Month': '2010_08',
    '2010.09 Month': '2010_09',
```

```
    '2010.10 Month': '2010_10',
    '2010.11 Month': '2010_11',
    '2010.12 Month': '2010_12',
    '2011.01 Month': '2011_01',
    '2011.02 Month': '2011_02',
    '2011.03 Month': '2011_03',
    '2011.04 Month': '2011_04',
    '2011.05 Month': '2011_05',
    '2011.06 Month': '2011_06',
    '2011.07 Month': '2011_07',
    '2011.08 Month': '2011_08',
    '2011.09 Month': '2011_09',
    '2011.10 Month': '2011_10',
    '2011.11 Month': '2011_11',
    '2011.12 Month': '2011_12',
    '2012.01 Month': '2012_01',
    '2012.02 Month': '2012_02',
    '2012.03 Month': '2012_03',
    '2012.04 Month': '2012_04',
    '2012.05 Month': '2012_05',
    '2012.06 Month': '2012_06',
    '2012.07 Month': '2012_07',
    '2012.08 Month': '2012_08',
    '2012.09 Month': '2012_09',
    '2012.10 Month': '2012_10',
    '2012.11 Month': '2012_11',
    '2012.12 Month': '2012_12',
    '2013.01 Month': '2013_01',
    '2013.02 Month': '2013_02',
    '2013.03 Month': '2013_03',
    '2013.04 Month': '2013_04',
    '2013.05 Month': '2013_05',
    '2013.06 Month': '2013_06',
    '2013.07 Month': '2013_07',
    '2013.08 Month': '2013_08',
    '2013.09 Month': '2013_09',
    '2013.10 Month': '2013_10',
    '2013.11 Month': '2013_11',
    '2013.12 Month': '2013_12',
    '2014.01 Month': '2014_01',
    '2014.02 Month': '2014_02',
    '2014.03 Month': '2014_03',
    '2014.04 Month': '2014_04',
    '2014.05 Month': '2014_05',
    '2014.06 Month': '2014_06',
    '2014.07 Month': '2014_07',
    '2014.08 Month': '2014_08',
```

```
    '2014.09 Month': '2014_09',
    '2014.10 Month': '2014_10',
    '2014.11 Month': '2014_11',
    '2014.12 Month': '2014_12',
    '2015.01 Month': '2015_01',
    '2015.02 Month': '2015_02',
    '2015.03 Month': '2015_03',
    '2015.04 Month': '2015_04',
    '2015.05 Month': '2015_05',
    '2015.06 Month': '2015_06',
    '2015.07 Month': '2015_07',
    '2015.08 Month': '2015_08',
    '2015.09 Month': '2015_09',
    '2015.10 Month': '2015_10',
    '2015.11 Month': '2015_11',
    '2015.12 Month': '2015_12',
    '2016.01 Month': '2016_01',
    '2016.02 Month': '2016_02',
    '2016.03 Month': '2016_03',
    '2016.04 Month': '2016_04',
    '2016.05 Month': '2016_05',
    '2016.06 Month': '2016_06',
    '2016.07 Month': '2016_07',
    '2016.08 Month': '2016_08',
    '2016.09 Month': '2016_09',
    '2016.10 Month': '2016_10',
    '2016.11 Month': '2016_11',
    '2016.12 Month': '2016_12',
    '2017.01 Month': '2017_01',
    '2017.02 Month': '2017_02',
    '2017.03 Month': '2017_03',
    '2017.04 Month': '2017_04',
    '2017.05 Month': '2017_05',
    '2017.06 Month': '2017_06',
    '2017.07 Month': '2017_07',
    '2017.08 Month': '2017_08',
    '2017.09 Month': '2017_09',
    '2017.10 Month': '2017_10',
    '2017.11 Month': '2017_11',
    '2017.12 Month': '2017_12',
    '2018.01 Month': '2018_01',
    '2018.02 Month': '2018_02',
    '2018.03 Month': '2018_03',
    '2018.04 Month': '2018_04',
    '2018.05 Month': '2018_05',
    '2018.06 Month': '2018_06',
    '2018.07 Month': '2018_07',
```

```
    '2018.08 Month': '2018_08',
    '2018.09 Month': '2018_09',
    '2018.10 Month': '2018_10',
    '2018.11 Month': '2018_11',
    '2018.12 Month': '2018_12',
    '2019.01 Month': '2019_01',
    '2019.02 Month': '2019_02',
    '2019.03 Month': '2019_03',
    '2019.04 Month': '2019_04',
    '2019.05 Month': '2019_05',
    '2019.06 Month': '2019_06',
    '2019.07 Month': '2019_07',
    '2019.08 Month': '2019_08',
    '2019.09 Month': '2019_09',
    '2019.10 Month': '2019_10',
    '2019.11 Month': '2019_11',
    '2019.12 Month': '2019_12',
    '2020.01 Month': '2020_01',
    '2020.02 Month': '2020_02',
    '2020.03 Month': '2020_03',
    '2020.04 Month': '2020_04',
    '2020.05 Month': '2020_05',
    '2020.06 Month': '2020_06',
    '2020.07 Month': '2020_07',
    '2020.08 Month': '2020_08',
    '2020.09 Month': '2020_09',
    '2020.10 Month': '2020_10',
    '2020.11 Month': '2020_11',
    '2020.12 Month': '2020_12',
    '2021.01 Month': '2021_01',
    '2021.02 Month': '2021_02',
    '2021.03 Month': '2021_03',
    '2021.04 Month': '2021_04',
    '2021.05 Month': '2021_05',
    '2021.06 Month': '2021_06',
    '2021.07 Month': '2021_07',
    '2021.08 Month': '2021_08',
    '2021.09 Month': '2021_09',
    '2021.10 Month': '2021_10',
    '2021.11 Month': '2021_11',
    '2021.12 Month': '2021_12',
    '2022.01 Month': '2022_01',
    '2022.02 Month': '2022_02',
    '2022.03 Month': '2022_03',
    '2022.04 Month': '2022_04',
    '2022.05 Month': '2022_05',
    '2022.06 Month': '2022_06',
```

```
    '2022.07 Month': '2022_07',
    '2022.08 Month': '2022_08',
    '2022.09 Month': '2022_09',
    '2022.10 Month': '2022_10',
    '2022.11 Month': '2022_11',
    '2022.12 Month': '2022_12',
    '2023.01 Month': '2023_01',
    '2023.02 Month': '2023_02',
    '2023.03 Month': '2023_03',
    '2023.04 Month': '2023_04',
    '2023.05 Month': '2023_05',
    '2023.06 Month': '2023_06',
    '2023.07 Month': '2023_07',
    '2023.08 Month': '2023_08',
    '2023.09 Month': '2023_09',
    '2023.10 Month': '2023_10',
    '2023.11 Month': '2023_11',
    '2023.12 Month': '2023_12',
    'Unnamed: 171': 'status_instrument'
    })
```

[149]:
```
# 06.02.20.03
# column rename to remove spaces and quotes
# dt08

dt08_summary_measurement_____00_rn = dt08_summary_measurement_____00.
 ↪rename(columns = {
    'Measurement date': 'date_measurement',
    'Station code': 'code_station',
    'Address': 'address',
    'Latitude': 'lat',
    'Longitude': 'lon',
    'SO2': 'so2',
    'NO2': 'no2',
    'O3': 'o3',
    'CO': 'co',
    'PM10': 'pm10',
    'PM2.5': 'pm2_5'
    })
```

[150]:
```
# 06.02.20.04
# column rename to remove spaces and quotes
# dt09

dt09_nat_emissions__________00_rn = dt09_nat_emissions__________00.
 ↪rename(columns = {
    'Division(1)': 'division',
```

```
    '1999': '1999_00',
    '1999.1': '1999_01',
    '1999.2': '1999_02',
    '1999.3': '1999_03',
    '1999.4': '1999_04',
    '1999.5': '1999_05',
    '1999.6': '1999_06',
    '2000': '2000_00',
    '2000.1': '2000_01',
    '2000.2': '2000_02',
    '2000.3': '2000_03',
    '2000.4': '2000_04',
    '2000.5': '2000_05',
    '2000.6': '2000_06',
    '2000.7': '2000_07',
    '2000.8': '2000_08',
    '2001': '2001_00',
    '2001.1': '2001_01',
    '2001.2': '2001_02',
    '2001.3': '2001_03',
    '2001.4': '2001_04',
    '2001.5': '2001_05',
    '2001.6': '2001_06',
    '2001.7': '2001_07',
    '2001.8': '2001_08',
    '2002': '2002_00',
    '2002.1': '2002_01',
    '2002.2': '2002_02',
    '2002.3': '2002_03',
    '2002.4': '2002_04',
    '2002.5': '2002_05',
    '2002.6': '2002_06',
    '2002.7': '2002_07',
    '2002.8': '2002_08',
    '2003': '2003_00',
    '2003.1': '2003_01',
    '2003.2': '2003_02',
    '2003.3': '2003_03',
    '2003.4': '2003_04',
    '2003.5': '2003_05',
    '2003.6': '2003_06',
    '2003.7': '2003_07',
    '2003.8': '2003_08',
    '2004': '2004_00',
    '2004.1': '2004_01',
    '2004.2': '2004_02',
    '2004.3': '2004_03',
```

```
    '2004.4': '2004_04',
    '2004.5': '2004_05',
    '2004.6': '2004_06',
    '2004.7': '2004_07',
    '2004.8': '2004_08',
    '2005': '2005_00',
    '2005.1': '2005_01',
    '2005.2': '2005_02',
    '2005.3': '2005_03',
    '2005.4': '2005_04',
    '2005.5': '2005_05',
    '2005.6': '2005_06',
    '2005.7': '2005_07',
    '2005.8': '2005_08',
    '2006': '2006_00',
    '2006.1': '2006_01',
    '2006.2': '2006_02',
    '2006.3': '2006_03',
    '2006.4': '2006_04',
    '2006.5': '2006_05',
    '2006.6': '2006_06',
    '2006.7': '2006_07',
    '2006.8': '2006_08',
    '2007': '2007_00',
    '2007.1': '2007_01',
    '2007.2': '2007_02',
    '2007.3': '2007_03',
    '2007.4': '2007_04',
    '2007.5': '2007_05',
    '2007.6': '2007_06',
    '2007.7': '2007_07',
    '2007.8': '2007_08',
    '2008': '2008_00',
    '2008.1': '2008_01',
    '2008.2': '2008_02',
    '2008.3': '2008_03',
    '2008.4': '2008_04',
    '2008.5': '2008_05',
    '2008.6': '2008_06',
    '2008.7': '2008_07',
    '2008.8': '2008_08',
    '2009': '2009_00',
    '2009.1': '2009_01',
    '2009.2': '2009_02',
    '2009.3': '2009_03',
    '2009.4': '2009_04',
    '2009.5': '2009_05',
```

```
    '2009.6': '2009_06',
    '2009.7': '2009_07',
    '2009.8': '2009_08',
    '2010': '2010_00',
    '2010.1': '2010_01',
    '2010.2': '2010_02',
    '2010.3': '2010_03',
    '2010.4': '2010_04',
    '2010.5': '2010_05',
    '2010.6': '2010_06',
    '2010.7': '2010_07',
    '2010.8': '2010_08',
    '2011': '2011_00',
    '2011.1': '2011_01',
    '2011.2': '2011_02',
    '2011.3': '2011_03',
    '2011.4': '2011_04',
    '2011.5': '2011_05',
    '2011.6': '2011_06',
    '2011.7': '2011_07',
    '2011.8': '2011_08',
    '2012': '2012_00',
    '2012.1': '2012_01',
    '2012.2': '2012_02',
    '2012.3': '2012_03',
    '2012.4': '2012_04',
    '2012.5': '2012_05',
    '2012.6': '2012_06',
    '2012.7': '2012_07',
    '2012.8': '2012_08',
    '2013': '2013_00',
    '2013.1': '2013_01',
    '2013.2': '2013_02',
    '2013.3': '2013_03',
    '2013.4': '2013_04',
    '2013.5': '2013_05',
    '2013.6': '2013_06',
    '2013.7': '2013_07',
    '2013.8': '2013_08',
    '2012.8': '2012_08',
    '2014': '2014_00',
    '2014.1': '2014_01',
    '2014.2': '2014_02',
    '2014.3': '2014_03',
    '2014.4': '2014_04',
    '2014.5': '2014_05',
    '2014.6': '2014_06',
```

```
    '2014.7': '2014_07',
    '2014.8': '2014_08',
    '2015': '2015_00',
    '2015.1': '2015_01',
    '2015.2': '2015_02',
    '2015.3': '2015_03',
    '2015.4': '2015_04',
    '2015.5': '2015_05',
    '2015.6': '2015_06',
    '2015.7': '2015_07',
    '2015.8': '2015_08',
    '2016': '2016_00',
    '2016.1': '2016_01',
    '2016.2': '2016_02',
    '2016.3': '2016_03',
    '2016.4': '2016_04',
    '2016.5': '2016_05',
    '2016.6': '2016_06',
    '2016.7': '2016_07',
    '2016.8': '2016_08',
    '2017': '2017_00',
    '2017.1': '2017_01',
    '2017.2': '2017_02',
    '2017.3': '2017_03',
    '2017.4': '2017_04',
    '2017.5': '2017_05',
    '2017.6': '2017_06',
    '2017.7': '2017_07',
    '2017.8': '2017_08',
    '2018': '2018_00',
    '2018.1': '2018_01',
    '2018.2': '2018_02',
    '2018.3': '2018_03',
    '2018.4': '2018_04',
    '2018.5': '2018_05',
    '2018.6': '2018_06',
    '2018.7': '2018_07',
    '2018.8': '2018_08',
    '2019': '2019_00',
    '2019.1': '2019_01',
    '2019.2': '2019_02',
    '2019.3': '2019_03',
    '2019.4': '2019_04',
    '2019.5': '2019_05',
    '2019.6': '2019_06',
    '2019.7': '2019_07',
    '2019.8': '2019_08',
```

```
    '2020': '2020_00',
    '2020.1': '2020_01',
    '2020.2': '2020_02',
    '2020.3': '2020_03',
    '2020.4': '2020_04',
    '2020.5': '2020_05',
    '2020.6': '2020_06',
    '2020.7': '2020_07',
    '2020.8': '2020_08',
    '2021': '2021_00',
    '2021.1': '2021_01',
    '2021.2': '2021_02',
    '2021.3': '2021_03',
    '2021.4': '2021_04',
    '2021.5': '2021_05',
    '2021.6': '2021_06',
    '2021.7': '2021_07',
    '2021.8': '2021_08'
    })
```

[151]:
```
# 06.02.20.05
# column rename to remove spaces and quotes
# dt10

dt10_seoul_air_____________00_rn = dt10_seoul_air_____________00.
 ↪rename(columns = {
    'dt': 'dt',
    'loc': 'loc',
    'lat': 'lat',
    'long': 'lon',
    'so2': 'so2',
    'no2': 'no2',
    'co': 'co',
    'o3': 'o3',
    'pm10': 'pm10',
    'pm2.5': 'pm2_5'
    })
```

[152]:
```
# 06.02.20.06
# column rename to remove spaces and quotes
# dt11

dt11_seoul_ave_air__________00_rn = dt11_seoul_ave_air__________00.
 ↪rename(columns = {
    '  ': '  ',
    '  ': '  ',
    '    (ppm)': '    _ppm',
```

```
    '  (ppm)': '  _ppm',
    '    (ppm)': '    _ppm',
    '   (ppm)': '   _ppm',
    '  ( / )': '  _m2',
    '   ( / )': '   _m2'
    })
```

[153]:
```
# 06.02.20.07
# confirm column names
# dt06-dt11

print(dt06_dt01_____________________00_rn.columns)
print('----------------------------------------')
print(dt07_dt02_____________________00_rn.columns)
print('----------------------------------------')
print(dt08_summary_measurement_____00_rn.columns)
print('----------------------------------------')
print(dt09_nat_emissions___________00_rn.columns)
print('----------------------------------------')
print(dt10_seoul_air_______________00_rn.columns)
print('----------------------------------------')
print(dt11_seoul_ave_air___________00_rn.columns)
```

```
Index(['classification_01', 'classification_02', 'item_01', 'item_02', 'unit',
       '2010_01', '2010_02', '2010_03', '2010_04', '2010_05',
       …
       '2023_04', '2023_05', '2023_06', '2023_07', '2023_08', '2023_09',
       '2023_10', '2023_11', '2023_12', 'status_instrument'],
      dtype='object', length=174)
----------------------------------------
Index(['classification', 'item', 'unit', '2010_01', '2010_02', '2010_03',
       '2010_04', '2010_05', '2010_06', '2010_07',
       …
       '2023_04', '2023_05', '2023_06', '2023_07', '2023_08', '2023_09',
       '2023_10', '2023_11', '2023_12', 'status_instrument'],
      dtype='object', length=172)
----------------------------------------
Index(['date_measurement', 'code_station', 'address', 'lat', 'lon', 'so2',
       'no2', 'o3', 'co', 'pm10', 'pm2_5'],
      dtype='object')
----------------------------------------
Index(['division', '1999_00', '1999_01', '1999_02', '1999_03', '1999_04',
       '1999_05', '1999_06', '2000_00', '2000_01',
       …
       '2020_08', '2021_00', '2021_01', '2021_02', '2021_03', '2021_04',
       '2021_05', '2021_06', '2021_07', '2021_08'],
      dtype='object', length=181)
----------------------------------------
```

```
Index(['dt', 'loc', 'lat', 'lon', 'so2', 'no2', 'co', 'o3', 'pm10', 'pm2_5'],
dtype='object')
----------------------------------------
Index(['   ', '   ', '    _ppm', '   _ppm', '    _ppm', '   _ppm',
       '   _m2', '   _m2'],
      dtype='object')
```

[154]:
```
# 06.02.21.01
# confirm types
# dt06

print(dt06_dt01_____________________00_rn['classification_01'].dtypes)
print(dt06_dt01_____________________00_rn['classification_02'].dtypes)
print(dt06_dt01_____________________00_rn['item_01'].dtypes)
print(dt06_dt01_____________________00_rn['item_02'].dtypes)
print(dt06_dt01_____________________00_rn['unit'].dtypes)
print(dt06_dt01_____________________00_rn['2010_01'].dtypes)
print(dt06_dt01_____________________00_rn['2023_12'].dtypes)
print(dt06_dt01_____________________00_rn['status_instrument'].dtypes)
```

```
object
object
object
object
float64
float64
object
float64
```

[155]:
```
# 06.02.21.02
# confirm types
# dt07

print(dt07_dt02_____________________00_rn['classification'].dtypes)
print(dt07_dt02_____________________00_rn['item'].dtypes)
print(dt07_dt02_____________________00_rn['unit'].dtypes)
print(dt07_dt02_____________________00_rn['2010_01'].dtypes)
print(dt07_dt02_____________________00_rn['2023_12'].dtypes)
print(dt07_dt02_____________________00_rn['status_instrument'].dtypes)
```

```
object
object
float64
float64
object
float64
```

[156]:
```
# 06.02.21.03
# confirm types
# dt08

print(dt08_summary_measurement_____00_rn['date_measurement'].dtypes)
print(dt08_summary_measurement_____00_rn['code_station'].dtypes)
print(dt08_summary_measurement_____00_rn['address'].dtypes)
print(dt08_summary_measurement_____00_rn['lat'].dtypes)
print(dt08_summary_measurement_____00_rn['lon'].dtypes)
print(dt08_summary_measurement_____00_rn['so2'].dtypes)
print(dt08_summary_measurement_____00_rn['so2'].dtypes)
print(dt08_summary_measurement_____00_rn['o3'].dtypes)
print(dt08_summary_measurement_____00_rn['co'].dtypes)
print(dt08_summary_measurement_____00_rn['pm10'].dtypes)
print(dt08_summary_measurement_____00_rn['pm2_5'].dtypes)
```

```
object
int64
object
float64
float64
float64
float64
float64
float64
float64
float64
```

[157]:
```
# 06.02.21.04
# confirm types
# dt09

print(dt09_nat_emissions__________00_rn['division'].dtypes)
print(dt09_nat_emissions__________00_rn['1999_00'].dtypes)
print(dt09_nat_emissions__________00_rn['2021_08'].dtypes)
```

```
object
object
object
```

[158]:
```
# 06.02.21.05
# confirm types
# dt10

print(dt10_seoul_air_____________00_rn['dt'].dtypes)
print(dt10_seoul_air_____________00_rn['loc'].dtypes)
print(dt10_seoul_air_____________00_rn['lat'].dtypes)
print(dt10_seoul_air_____________00_rn['lon'].dtypes)
```

```
print(dt10_seoul_air_____________00_rn['so2'].dtypes)
print(dt10_seoul_air_____________00_rn['no2'].dtypes)
print(dt10_seoul_air_____________00_rn['co'].dtypes)
print(dt10_seoul_air_____________00_rn['o3'].dtypes)
print(dt10_seoul_air_____________00_rn['pm10'].dtypes)
print(dt10_seoul_air_____________00_rn['pm2_5'].dtypes)
```

```
int64
int64
float64
float64
float64
float64
float64
float64
float64
float64
```

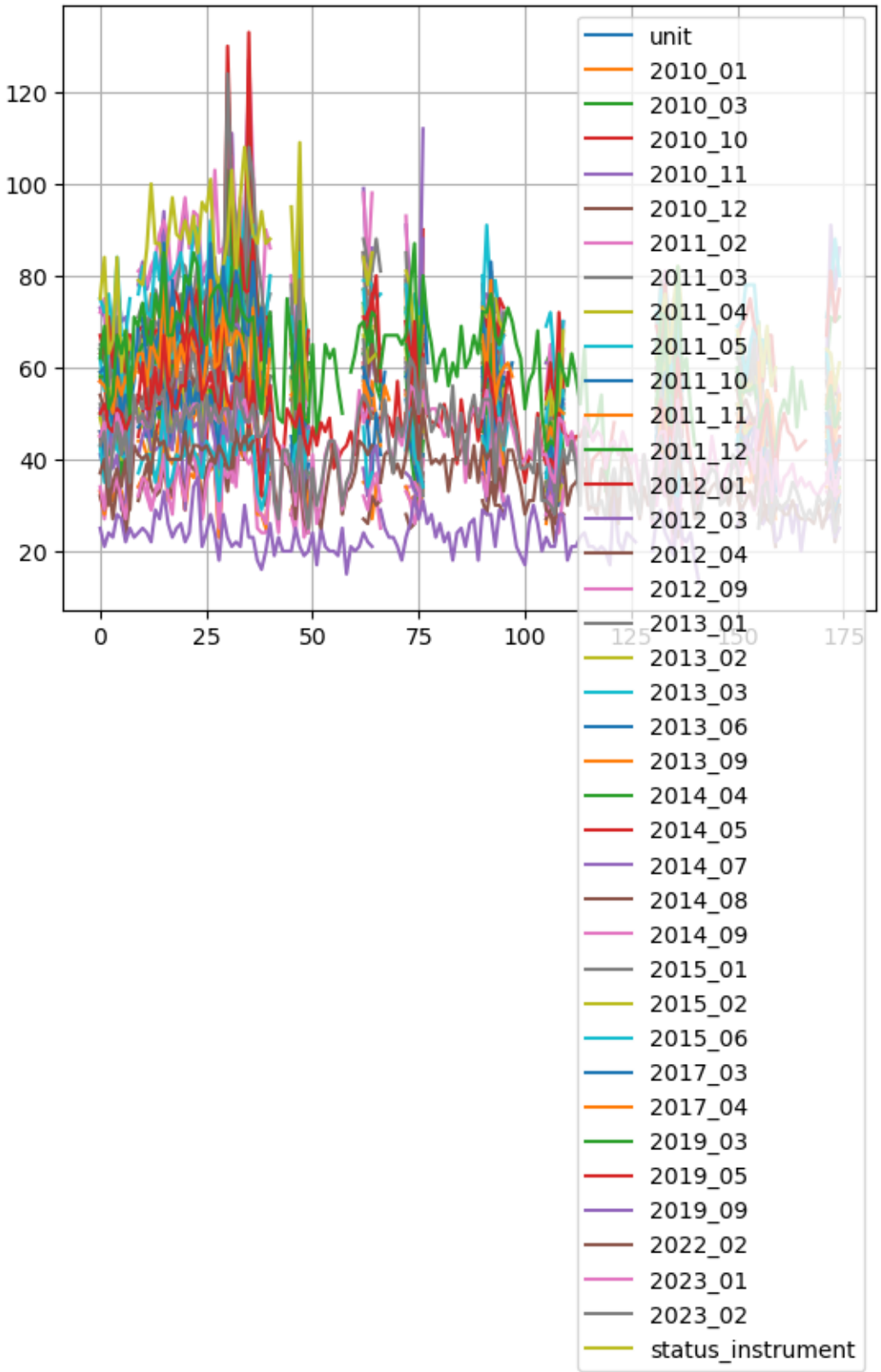
[159]:
```
# 06.02.21.06
# confirm types
# dt11

print(dt11_seoul_ave_air__________00_rn['  '].dtypes)
print(dt11_seoul_ave_air__________00_rn['  '].dtypes)
print(dt11_seoul_ave_air__________00_rn['    _ppm'].dtypes)
print(dt11_seoul_ave_air__________00_rn['  _ppm'].dtypes)
print(dt11_seoul_ave_air__________00_rn['    _ppm'].dtypes)
print(dt11_seoul_ave_air__________00_rn['   _ppm'].dtypes)
print(dt11_seoul_ave_air__________00_rn['  _m2'].dtypes)
print(dt11_seoul_ave_air__________00_rn['   _m2'].dtypes)
```

```
int64
object
float64
float64
float64
float64
float64
float64
```

[160]:
```
# 06.02.22.01
# return basic plot
# rendered basic plot to see visually and to determine if data is numeric
# dt06

plt.figure(figsize=(18,8))
dt06_dt01___________________00_rn.plot()
plt.box(True)
```

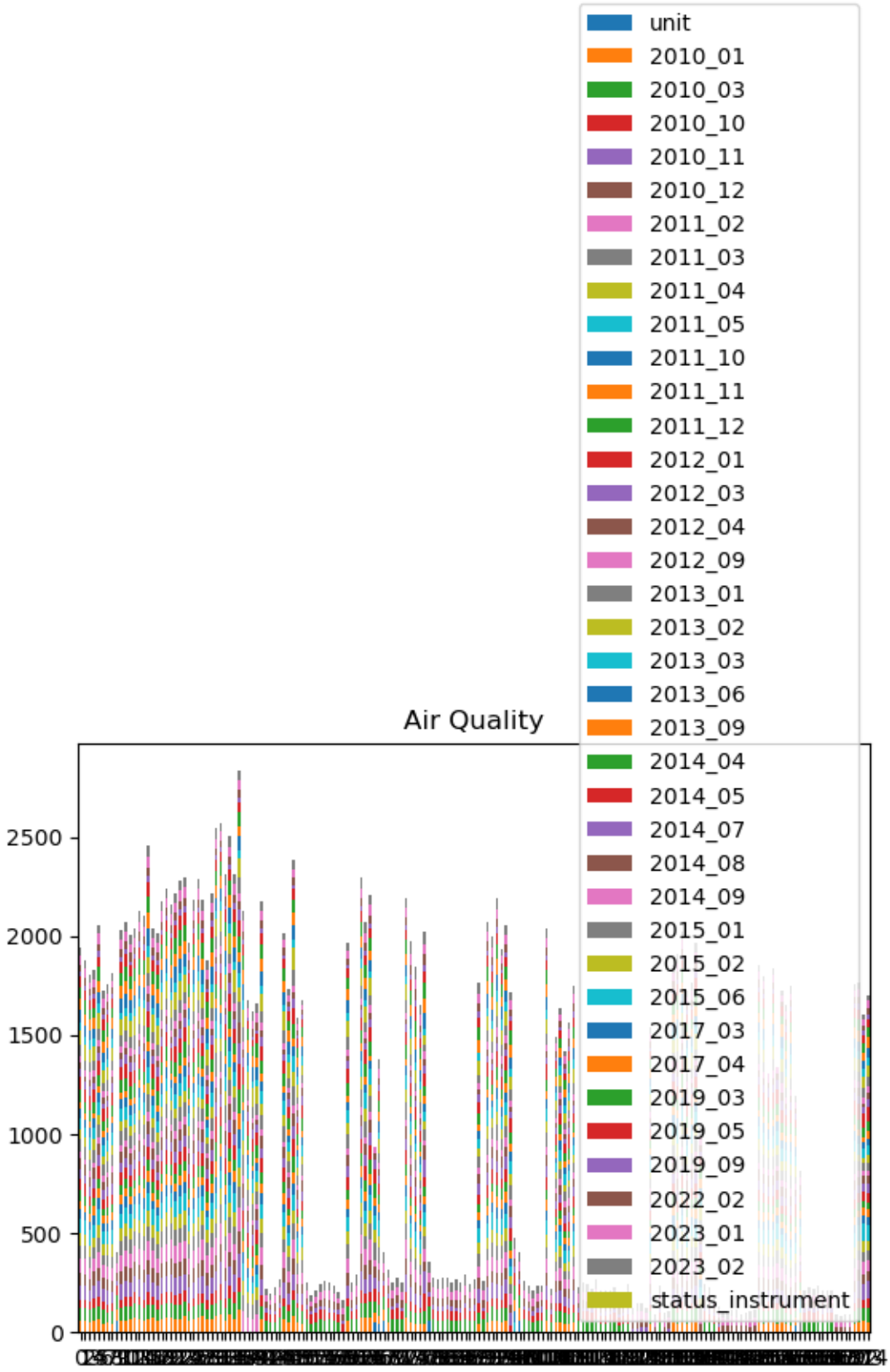
```
plt.grid(True)
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```


120
100
80
60
40
20
0
25
50
75
100
125
150
175
unit
2010_01
2010_03
2010_10
2010_11
2010_12
2011_02
2011_03
2011_04
2011_05
2011_10
2011_11
2011_12
2012_01
2012_03
2012_04
2012_09
2013_01
2013_02
2013_03
2013_06
2013_09
2014_04
2014_05
2014_07
2014_08
2014_09
2015_01
2015_02
2015_06
2017_03
2017_04
2019_03
2019_05
2019_09
2022_02
2023_01
2023_02
status_instrument

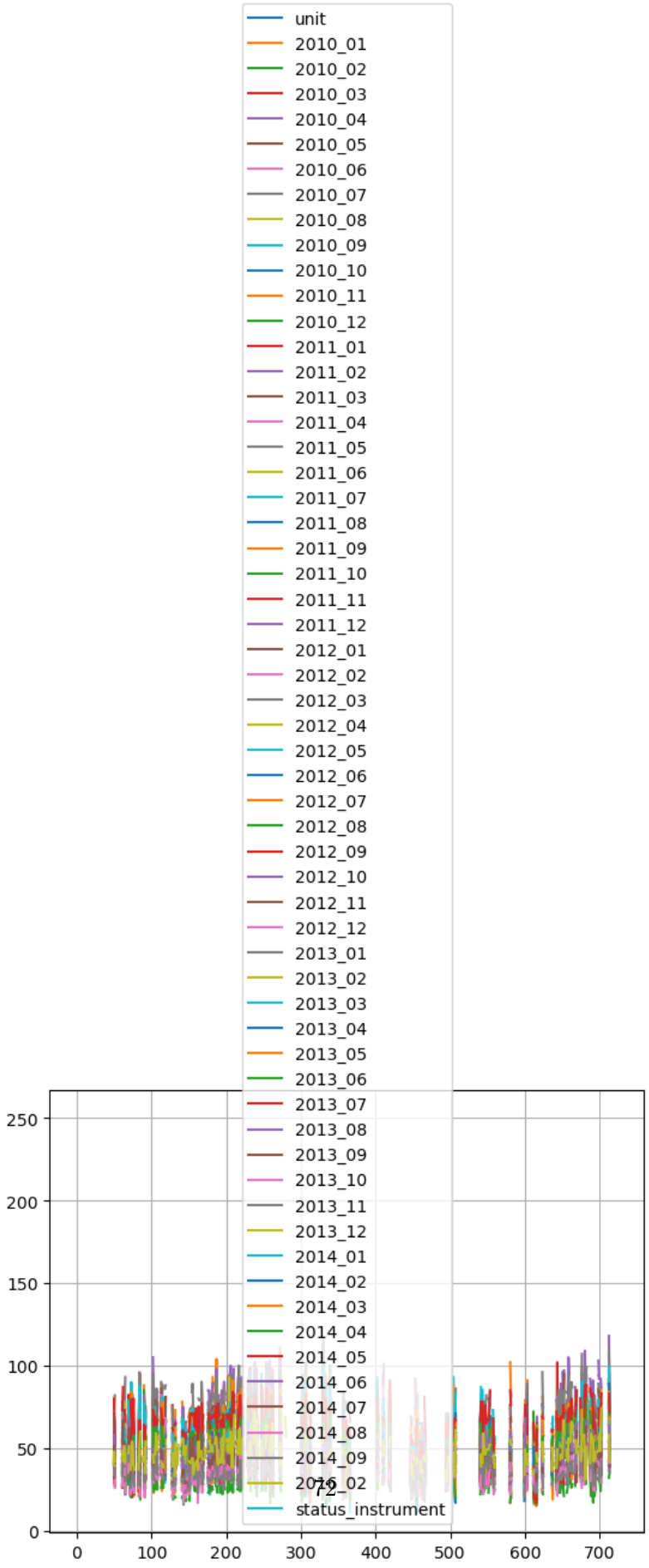
[161]:
```
# 06.02.22.02
# render bar chart
# dt06

plt.figure(figsize=(18,8))
dt06_dt01____________________00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```


Air Quality
2500
2000
1500
1000
500
0
unit
2010_01
2010_03
2010_10
2010_11
2010_12
2011_02
2011_03
2011_04
2011_05
2011_10
2011_11
2011_12
2012_01
2012_03
2012_04
2012_09
2013_01
2013_02
2013_03
2013_06
2013_09
2014_04
2014_05
2014_07
2014_08
2014_09
2015_01
2015_02
2015_06
2017_03
2017_04
2019_03
2019_05
2019_09
2022_02
2023_01
2023_02
status_instrument

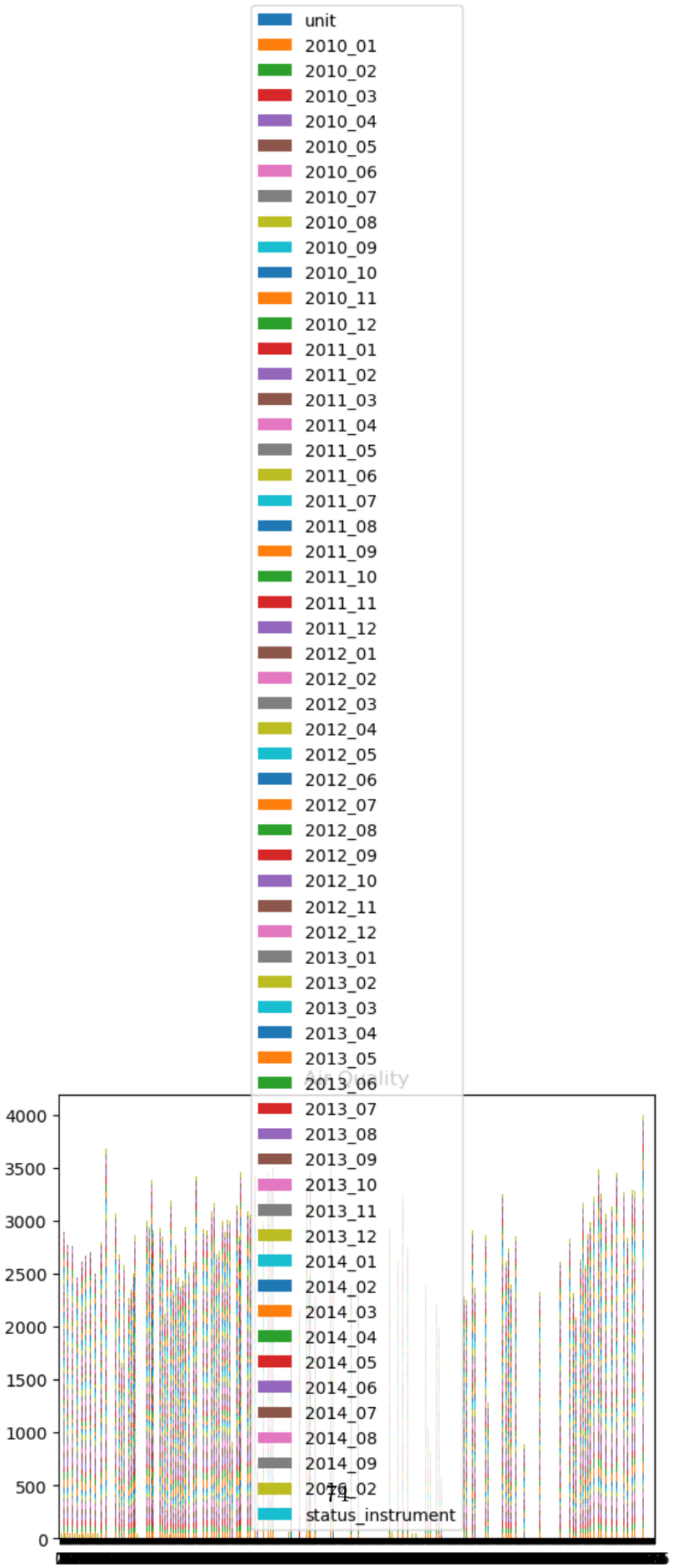
[162]:
```
# 06.02.22.03
# return basic plot
# rendered basic plot to see visually and to determine if data is numeric
# dt07

plt.figure(figsize=(18,8))
dt07_dt02_____________________00_rn.plot()
plt.box(True)
plt.grid(True)
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```


unit
2010_01
2010_02
2010_03
2010_04
2010_05
2010_06
2010_07
2010_08
2010_09
2010_10
2010_11
2010_12
2011_01
2011_02
2011_03
2011_04
2011_05
2011_06
2011_07
2011_08
2011_09
2011_10
2011_11
2011_12
2012_01
2012_02
2012_03
2012_04
2012_05
2012_06
2012_07
2012_08
2012_09
2012_10
2012_11
2012_12
2013_01
2013_02
2013_03
2013_04
2013_05
2013_06
2013_07
2013_08
2013_09
2013_10
2013_11
2013_12
2014_01
2014_02
2014_03
2014_04
2014_05
2014_06
2014_07
2014_08
2014_09
2014_10
2014_11
2014_12
2015_01
2015_02
status_instrument
250
200
150
100
50
0
0
100
200
300
400
500
600
700

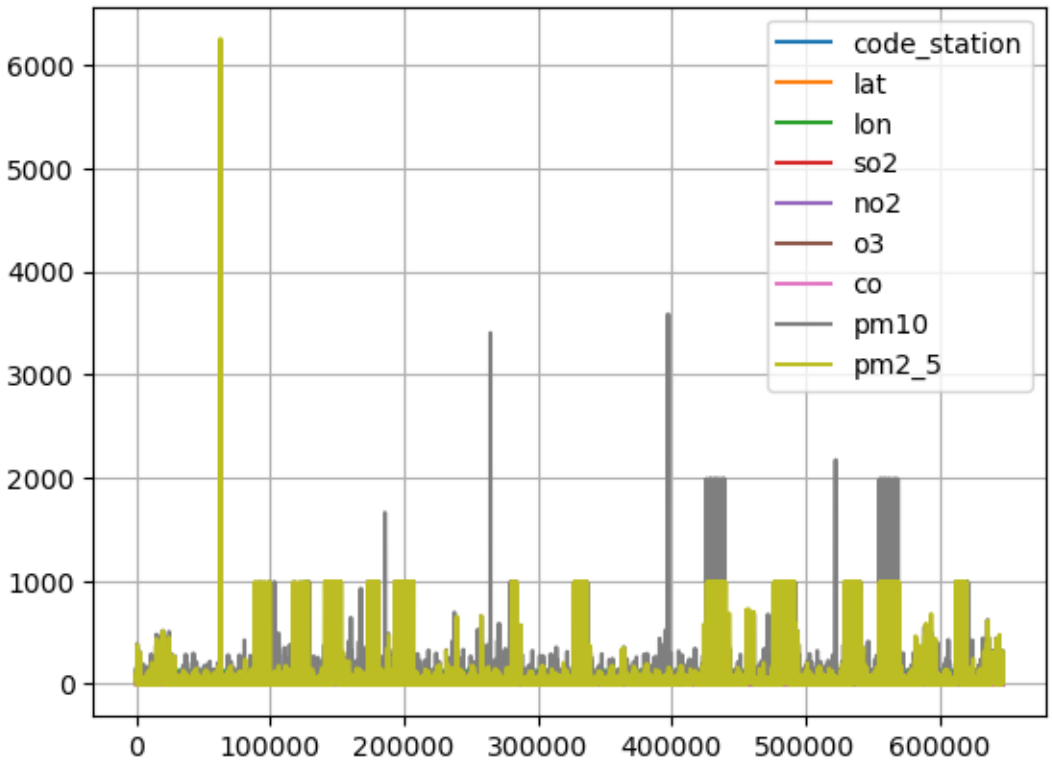
[163]:
```
# 06.02.22.04
# render bar chart
# dt07

plt.figure(figsize=(18,8))
dt07_dt02_____________________00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```


unit
2010_01
2010_02
2010_03
2010_04
2010_05
2010_06
2010_07
2010_08
2010_09
2010_10
2010_11
2010_12
2011_01
2011_02
2011_03
2011_04
2011_05
2011_06
2011_07
2011_08
2011_09
2011_10
2011_11
2011_12
2012_01
2012_02
2012_03
2012_04
2012_05
2012_06
2012_07
2012_08
2012_09
2012_10
2012_11
2012_12
2013_01
2013_02
2013_03
2013_04
2013_05
2013_06
2013_07
2013_08
2013_09
2013_10
2013_11
2013_12
2014_01
2014_02
2014_03
2014_04
2014_05
2014_06
2014_07
2014_08
2014_09
2014_10
status_instrument
4000
3500
3000
2500
2000
1500
1000
500
0

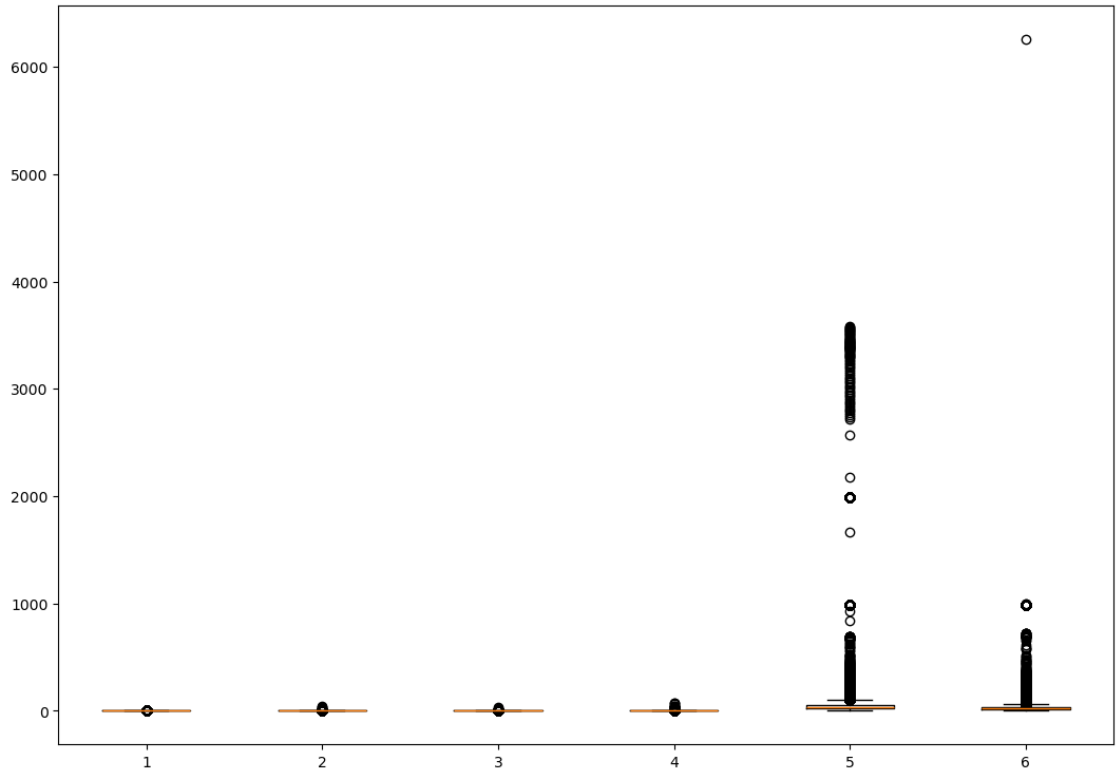
[164]:
```
# 06.02.22.05
# return basic plot
# rendered basic plot to see visually and to determine if data is numeric
# dt08

plt.figure(figsize=(18,8))
dt08_summary_measurement_____00_rn.plot()
plt.box(True)
plt.grid(True)
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```



[165]:
```
# 06.02.22.06
# boxplot
# dt08

```

```
np.random.seed(10)
dt08_summary_measurement_____00_rn_boxplot =␣
 ↪[dt08_summary_measurement_____00_rn['so2'],␣
 ↪dt08_summary_measurement_____00_rn['no2'],␣
 ↪dt08_summary_measurement_____00_rn['o3'],␣
 ↪dt08_summary_measurement_____00_rn['co'],␣
 ↪dt08_summary_measurement_____00_rn['pm10'],␣
 ↪dt08_summary_measurement_____00_rn['pm2_5']]
fig = plt.figure(figsize =(10, 7))
ax = fig.add_axes([0, 0, 1, 1])
bp = ax.boxplot(dt08_summary_measurement_____00_rn_boxplot)
plt.show()
```



[166]:
```
# 06.02.22.07
# return basic plot
# rendered basic plot to see visually and to determine if data is numeric
# dt09

'''plt.figure(figsize=(18,8))
dt09_nat_emissions__________00_rn.plot()
plt.box(True)
plt.grid(True)
```

```
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()'''
```

[166]:
```
"plt.figure(figsize=(18,8))\ndt09_nat_emissions__________00_rn.plot()\nplt.box(
True)\nplt.grid(True)\nplt.title('', fontsize = 16, color =
'#0047ab')\nplt.xlabel('', fontsize = 14, color = '#0047ab')\nplt.ylabel('',
fontsize = 14, color = '#0047ab')\nplt.show()"
```

[167]:
```
# 06.02.22.08
# render bar chart
# dt09

'''plt.figure(figsize=(18,8))
dt09_nat_emissions__________00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()'''
```

[167]:
```
"plt.figure(figsize=(18,8))\ndt09_nat_emissions__________00_rn.plot(kind='bar',
stacked=True)\nplt.title('Air Quality')\nplt.xticks(rotation=0,
ha='center')\nplt.show()"
```

[168]:
```
# 06.02.22.09
# return basic plot
# rendered basic plot to see visually and to determine if data is numeric
# dt10

'''plt.figure(figsize=(18,8))
dt10_seoul_air_____________00_rn.plot()
plt.box(True)
plt.grid(True)
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()'''
```

[168]:
```
"plt.figure(figsize=(18,8))\ndt10_seoul_air_____________00_rn.plot()\nplt.box(
True)\nplt.grid(True)\nplt.title('', fontsize = 16, color =
'#0047ab')\nplt.xlabel('', fontsize = 14, color = '#0047ab')\nplt.ylabel('',
fontsize = 14, color = '#0047ab')\nplt.show()"
```

[169]:
```
# 06.02.22.10
# render bar chart
# dt10
```

```
'''plt.figure(figsize=(18,8))
dt10_seoul_air_____________00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()'''
```

[169]:
```
"plt.figure(figsize=(18,8))\ndt10_seoul_air_____________00_rn.plot(kind='bar',
stacked=True)\nplt.title('Air Quality')\nplt.xticks(rotation=0,
ha='center')\nplt.show()"
```

[170]:
```
# 06.02.22.11
# return basic plot
# rendered basic plot to see visually and to determine if data is numeric
# dt11

'''plt.figure(figsize=(18,8))
dt11_seoul_ave_air__________00_rn.plot()
plt.box(True)
plt.grid(True)
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()'''
```

[170]:
```
"plt.figure(figsize=(18,8))\ndt11_seoul_ave_air__________00_rn.plot()\nplt.box(
True)\nplt.grid(True)\nplt.title('', fontsize = 16, color =
'#0047ab')\nplt.xlabel('', fontsize = 14, color = '#0047ab')\nplt.ylabel('',
fontsize = 14, color = '#0047ab')\nplt.show()"
```

[171]:
```
# 06.02.22.12
# render bar chart
# dt11

'''plt.figure(figsize=(18,8))
dt11_seoul_ave_air__________00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()'''
```

[171]:
```
"plt.figure(figsize=(18,8))\ndt11_seoul_ave_air__________00_rn.plot(kind='bar',
stacked=True)\nplt.title('Air Quality')\nplt.xticks(rotation=0,
ha='center')\nplt.show()"
```

[172]:
```
# 06.02.23.01
# read csv
# assign variable
# dt12
```

```
dt12_streets_____________00 = pd.read_csv('116_DT_MLTM_962_20240712093545.
 ↪csv')
```

[173]:
```
# 06.02.23.02
# read csv
# assign variable
# dt13

dt13_ave_traffic_____________00 = pd.
 ↪read_csv('Average_Daily_Traffic_by_Road_and_Vehicle_Types_20240712093400.
 ↪csv')
```

[174]:
```
# 06.02.23.03
# read csv
# assign variable
# dt14

dt14_yearly_____________00 = pd.read_csv('Yearly_ADT_20240712093451.csv')
```

[175]:
```
# 06.02.24.01
# return first and last ten rows
# dt012

print(dt12_streets_____________00.head(10))
print(dt12_streets_____________00.tail(10))
```

```
  Classification              Level01  …     2023 Year  Unnamed: 23
0          Total                Total  …  1.158776e+08          NaN
1          Total                Total  …  1.071491e+08          NaN
2          Total                Total  …  1.022051e+08          NaN
3          Total                Total  …  9.539000e+01          NaN
4          Total                Total  …  4.944024e+06          NaN
5          Total                Total  …  4.974020e+05          NaN
6          Total                Total  …  8.231110e+06          NaN
7          Total  National Expressway  …  4.972710e+06          NaN
8          Total  National Expressway  …  4.972710e+06          NaN
9          Total     National Highway  …  1.412458e+07          NaN

[10 rows x 24 columns]
    Classification   Level01            Item  …  2022 Year  2023 Year
Unnamed: 23
908           Jeju  Gun Road        Unopened  …     9370.0     9370.0
NaN
909           Jeju  Gun Road        Unopened  …   189270.0   189270.0
NaN
910           Jeju  Gun Road     Gun highway  …   887600.0   887600.0
NaN
```

```
911           Jeju   Gu Road         Opening  …        0.0        0.0
NaN
912           Jeju   Gu Road        Pavement  …        0.0        NaN
NaN
913           Jeju   Gu Road  Pavement Ratio  …        0.0        NaN
NaN
914           Jeju   Gu Road         Unpaved  …        0.0        NaN
NaN
915           Jeju   Gu Road        Unopened  …        0.0        NaN
NaN
916           Jeju   Gu Road        Unopened  …        0.0        NaN
NaN
917           Jeju   Gu Road      Gu highway  …        0.0        0.0
NaN

[10 rows x 24 columns]
```

[176]:
```
# 06.02.24.02
# return first and last ten rows
# dt013

print(dt13_ave_traffic_____________00.head(10))
print(dt13_ave_traffic_____________00.tail(10))
```

```
                  Road Type(1)  …                                2023.2
0                 Road Type(1)  …  Rate of increase compared to year ago
1                   Expressway  …                                   0.8
2                   Expressway  …                                   0.9
3                   Expressway  …                                   5.4
4                   Expressway  …                                   0.4
5               National Route  …                                  -0.6
6               National Route  …                                  -0.5
7               National Route  …                                   1.1
8               National Route  …                                  -1.1
9  Local Road(support of state)  …                                  -4.3

[10 rows x 62 columns]
                   Road Type(1) Vehicle Type(1)  2004  …  2023 2023.1 2023.2
7                National Route             Bus   350  …   191    1.4    1.1
8                National Route    freight car   3447  …  2794   21.2   -1.1
9   Local Road(support of state)           Total  9063  …  8096  100.0   -4.3
10  Local Road(support of state)           sedan  5550  …  6038   74.6   -3.5
11  Local Road(support of state)             Bus  1067  …   168    2.1   -6.1
12  Local Road(support of state)    freight car   2446  …  1890   23.3   -6.5
13                   Local Road           Total  4444  …  5277  100.0   -1.2
14                   Local Road           sedan  2505  …  3887   73.6   -1.5
15                   Local Road             Bus   516  …   120    2.3    0.0
16                   Local Road    freight car   1423  …  1270   24.1   -0.4
```

```
[10 rows x 62 columns]
```

[177]:
```
# 06.02.24.03
# return first and last ten rows
# dt014

print(dt14_yearly__________________00.head(10))
print(dt14_yearly__________________00.tail(10))
```

```
   Road Type(1)                    Classification(1)  …     2023 2023.1
0  Road Type(1)                    Classification(1)  …   Length  Ratio
1    Expressway                                Total  …   4966.5  100.0
2    Expressway                Average Daily Traffic  …  52544.0    0.0
3    Expressway  Average Daily Traffic (vehicle/day)  …        -      -
4    Expressway                           0-10,000  …    133.2    2.7
5    Expressway                         0 - 10,000  …        -      -
6    Expressway                      10,001 - 40,000  …        -      -
7    Expressway                        10,001-40,000  …   2346.9   47.3
8    Expressway                        40,001-70,000  …   1436.8   28.9
9    Expressway                      40,001 - 70,000  …        -      -

[10 rows x 44 columns]
   Road Type(1) Classification(1)    2003 2003.1  …    2022 2022.1    2023
2023.1
57   Local Road     1,001 - 3,000  4405.3   41.3  …       -      -       -
-
58   Local Road     3,001 - 5,000  1978.8   18.6  …       -      -       -
-
59   Local Road       3,001-5,000       -      -  …  1746.2   15.4  1561.0
13.9
60   Local Road     5,001 - 7,000  1026.9    9.6  …       -      -       -
-
61   Local Road       5,001-7,000       -      -  …   942.1    8.3   939.5
8.3
62   Local Road       7,001-9,000       -      -  …   489.2    4.3   501.0
4.5
63   Local Road     7,001 - 9,000   526.8    4.9  …       -      -       -
-
64   Local Road    9,001 - 11,000   255.2    2.4  …       -      -       -
-
65   Local Road      9,001-11,000       -      -  …   349.7    3.1   278.7
2.5
66   Local Road       Over 11,001   829.4    7.8  …  1247.9   11.0  1273.8
11.3

[10 rows x 44 columns]
```

[178]:
```
# 06.02.25.01
# return dimensions
# dt12-dt14

print(dt12_streets_____________00.shape)
print('-----------------------------------')
print(dt13_ave_traffic_____________00.shape)
print('-----------------------------------')
print(dt14_yearly_____________00.shape)
```

```
(918, 24)
-----------------------------------
(17, 62)
-----------------------------------
(67, 44)
```

[179]:
```
# 06.02.25.02
# confirm column names
# dt12-dt14

print(dt12_streets_____________00.columns)
print('-------------------------------------')
print(dt13_ave_traffic_____________00.columns)
print('-------------------------------------')
print(dt14_yearly_____________00.columns)
```

```
Index(['Classification', 'Level01', 'Item', 'UNIT', '2005 Year', '2006 Year',
       '2007 Year', '2008 Year', '2009 Year', '2010 Year', '2011 Year',
       '2012 Year', '2013 Year', '2014 Year', '2015 Year', '2016 Year',
       '2017 Year', '2018 Year', '2019 Year', '2020 Year', '2021 Year',
       '2022 Year', '2023 Year', 'Unnamed: 23'],
      dtype='object')
-------------------------------------
Index(['Road Type(1)', 'Vehicle Type(1)', '2004', '2004.1', '2004.2', '2005',
       '2005.1', '2005.2', '2006', '2006.1', '2006.2', '2007', '2007.1',
       '2007.2', '2008', '2008.1', '2008.2', '2009', '2009.1', '2009.2',
       '2010', '2010.1', '2010.2', '2011', '2011.1', '2011.2', '2012',
       '2012.1', '2012.2', '2013', '2013.1', '2013.2', '2014', '2014.1',
       '2014.2', '2015', '2015.1', '2015.2', '2016', '2016.1', '2016.2',
       '2017', '2017.1', '2017.2', '2018', '2018.1', '2018.2', '2019',
       '2019.1', '2019.2', '2020', '2020.1', '2020.2', '2021', '2021.1',
       '2021.2', '2022', '2022.1', '2022.2', '2023', '2023.1', '2023.2'],
      dtype='object')
-------------------------------------
Index(['Road Type(1)', 'Classification(1)', '2003', '2003.1', '2004', '2004.1',
       '2005', '2005.1', '2006', '2006.1', '2007', '2007.1', '2008', '2008.1',
       '2009', '2009.1', '2010', '2010.1', '2011', '2011.1', '2012', '2012.1',
       '2013', '2013.1', '2014', '2014.1', '2015', '2015.1', '2016', '2016.1',
```

```
       '2017', '2017.1', '2018', '2018.1', '2019', '2019.1', '2020', '2020.1',
       '2021', '2021.1', '2022', '2022.1', '2023', '2023.1'],
      dtype='object')
```

[180]:
```
# 06.02.26.01
# column rename to remove spaces and quotes
# dt12

dt12_streets_____________00_rn = dt12_streets_____________00.
 ↪rename(columns = {
    'Classification': 'classification',
    'Level01': 'level',
    'Item': 'item',
    'UNIT': 'unit',
    '2005 Year': '2005',
    '2006 Year': '2006',
    '2007 Year': '2007',
    '2008 Year': '2008',
    '2009 Year': '2009',
    '2010 Year': '2010',
    '2011 Year': '2011',
    '2012 Year': '2012',
    '2013 Year': '2013',
    '2014 Year': '2014',
    '2015 Year': '2015',
    '2016 Year': '2016',
    '2017 Year': '2017',
    '2018 Year': '2018',
    '2019 Year': '2019',
    '2020 Year': '2020',
    '2021 Year': '2021',
    '2022 Year': '2022',
    '2023 Year': '2023',
    'Unnamed: 23': 'unnamed_23'
    })
```

[181]:
```
# 06.02.26.02
# column rename to remove spaces and quotes
# dt13

dt13_ave_traffic___________00_rn = dt13_ave_traffic___________00.
 ↪rename(columns = {
    'Road Type(1)': 'type_road',
    'Vehicle Type(1)': 'type_vehicle',
    '2004': '2004_00',
    '2004.1': '2004_01',
    '2004.2': '2004_02',
```

```
    '2005': '2005_00',
    '2005.1': '2005_01',
    '2005.2': '2005_02',
    '2006': '2006_00',
    '2006.1': '2006_01',
    '2006.2': '2006_02',
    '2007': '2007_00',
    '2007.1': '2007_01',
    '2007.2': '2007_02',
    '2008': '2008_00',
    '2008.1': '2008_01',
    '2008.2': '2008_02',
    '2009': '2009_00',
    '2009.1': '2009_01',
    '2009.2': '2009_02',
    '2010': '2010_00',
    '2010.1': '2010_01',
    '2010.2': '2010_02',
    '2011': '2011_00',
    '2011.1': '2011_01',
    '2011.2': '2011_02',
    '2012': '2012_00',
    '2012.1': '2012_01',
    '2012.2': '2012_02',
    '2013': '2013_00',
    '2013.1': '2013_01',
    '2013.2': '2013_02',
    '2014': '2014_00',
    '2014.1': '2014_01',
    '2014.2': '2014_02',
    '2015': '2015_00',
    '2015.1': '2015_01',
    '2015.2': '2015_02',
    '2016': '2016_00',
    '2016.1': '2016_01',
    '2016.2': '2016_02',
    '2017': '2017_00',
    '2017.1': '2017_01',
    '2017.2': '2017_02',
    '2018': '2018_00',
    '2018.1': '2018_01',
    '2018.2': '2018_02',
    '2019': '2019_00',
    '2019.1': '2019_01',
    '2019.2': '2019_02',
    '2020': '2020_00',
    '2020.1': '2020_01',
```

```
    '2020.2': '2020_02',
    '2021': '2021_00',
    '2021.1': '2021_01',
    '2021.2': '2021_02',
    '2022': '2022_00',
    '2022.1': '2022_01',
    '2022.2': '2022_02',
    '2023': '2023_00',
    '2023.1': '2023_01',
    '2023.2': '2023_02'
    })
```

[182]:
```
# 06.02.26.03
# column rename to remove spaces and quotes
# dt14

dt14_yearly__________________00_rn = dt14_yearly__________________00.
 ↪rename(columns = {
    'Road Type(1)': 'type_road',
    'Classification(1)': 'classification',
    '2003': '2003_00',
    '2003.1': '2003_01',
    '2004': '2004_00',
    '2004.1': '2004_01',
    '2005': '2005_00',
    '2005.1': '2005_01',
    '2006': '2006_00',
    '2006.1': '2006_01',
    '2007': '2007_00',
    '2007.1': '2007_01',
    '2008': '2008_00',
    '2008.1': '2008_01',
    '2009': '2009_00',
    '2009.1': '2009_01',
    '2010': '2010_00',
    '2010.1': '2010_01',
    '2011': '2011_00',
    '2011.1': '2011_01',
    '2012': '2012_00',
    '2012.1': '2012_01',
    '2013': '2013_00',
    '2013.1': '2013_01',
    '2014': '2014_00',
    '2014.1': '2014_01',
    '2015': '2015_00',
    '2015.1': '2015_01',
    '2016': '2016_00',
```

```
    '2016.1': '2016_01',
    '2017': '2017_00',
    '2017.1': '2017_01',
    '2018': '2018_00',
    '2018.1': '2018_01',
    '2019': '2019_00',
    '2019.1': '2019_01',
    '2020': '2020_00',
    '2020.1': '2020_01',
    '2021': '2021_00',
    '2021.1': '2021_01',
    '2022': '2022_00',
    '2022.1': '2022_01',
    '2023': '2023_00',
    '2023.1': '2023_01'
    })
```

[183]:
```
# 06.02.26.04
# confirm column names
# dt12-dt14

print(dt12_streets_____________00_rn.columns)
print('-------------------------------------')
print(dt13_ave_traffic_____________00_rn.columns)
print('-------------------------------------')
print(dt14_yearly_____________00_rn.columns)
```

```
Index(['classification', 'level', 'item', 'unit', '2005', '2006', '2007',
       '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015', '2016',
       '2017', '2018', '2019', '2020', '2021', '2022', '2023', 'unnamed_23'],
      dtype='object')
-------------------------------------
Index(['type_road', 'type_vehicle', '2004_00', '2004_01', '2004_02', '2005_00',
       '2005_01', '2005_02', '2006_00', '2006_01', '2006_02', '2007_00',
       '2007_01', '2007_02', '2008_00', '2008_01', '2008_02', '2009_00',
       '2009_01', '2009_02', '2010_00', '2010_01', '2010_02', '2011_00',
       '2011_01', '2011_02', '2012_00', '2012_01', '2012_02', '2013_00',
       '2013_01', '2013_02', '2014_00', '2014_01', '2014_02', '2015_00',
       '2015_01', '2015_02', '2016_00', '2016_01', '2016_02', '2017_00',
       '2017_01', '2017_02', '2018_00', '2018_01', '2018_02', '2019_00',
       '2019_01', '2019_02', '2020_00', '2020_01', '2020_02', '2021_00',
       '2021_01', '2021_02', '2022_00', '2022_01', '2022_02', '2023_00',
       '2023_01', '2023_02'],
      dtype='object')
-------------------------------------
Index(['type_road', 'classification', '2003_00', '2003_01', '2004_00',
       '2004_01', '2005_00', '2005_01', '2006_00', '2006_01', '2007_00',
       '2007_01', '2008_00', '2008_01', '2009_00', '2009_01', '2010_00',
```

```
       '2010_01', '2011_00', '2011_01', '2012_00', '2012_01', '2013_00',
       '2013_01', '2014_00', '2014_01', '2015_00', '2015_01', '2016_00',
       '2016_01', '2017_00', '2017_01', '2018_00', '2018_01', '2019_00',
       '2019_01', '2020_00', '2020_01', '2021_00', '2021_01', '2022_00',
       '2022_01', '2023_00', '2023_01'],
      dtype='object')
```

[184]:
```
# 06.02.27.01
# confirm types
# dt12

print(dt12_streets_____________00_rn['classification'].dtypes)
print(dt12_streets_____________00_rn['level'].dtypes)
print(dt12_streets_____________00_rn['item'].dtypes)
print(dt12_streets_____________00_rn['2005'].dtypes)
print(dt12_streets_____________00_rn['2023'].dtypes)
print(dt12_streets_____________00_rn['unnamed_23'].dtypes)
```

```
object
object
object
float64
float64
float64
```

[185]:
```
# 06.02.27.02
# confirm types
# dt13

print(dt13_ave_traffic_____________00_rn['type_road'].dtypes)
print(dt13_ave_traffic_____________00_rn['type_vehicle'].dtypes)
print(dt13_ave_traffic_____________00_rn['2004_00'].dtypes)
print(dt13_ave_traffic_____________00_rn['2023_02'].dtypes)
```

```
object
object
object
object
```

[186]:
```
# 06.02.27.03
# confirm types
# dt14

print(dt14_yearly_____________00_rn['type_road'].dtypes)
print(dt14_yearly_____________00_rn['classification'].dtypes)
print(dt14_yearly_____________00_rn['2003_00'].dtypes)
print(dt14_yearly_____________00_rn['2023_01'].dtypes)
```

```
object
```

```
object
object
object
```

[187]:
```
# 06.02.28.01
# change types
# dt12

dt12_streets_____________00_rn['2005'].astype('float')
dt12_streets_____________00_rn['2006'].astype('float')
dt12_streets_____________00_rn['2007'].astype('float')
dt12_streets_____________00_rn['2008'].astype('float')
dt12_streets_____________00_rn['2009'].astype('float')
dt12_streets_____________00_rn['2010'].astype('float')
dt12_streets_____________00_rn['2011'].astype('float')
dt12_streets_____________00_rn['2012'].astype('float')
dt12_streets_____________00_rn['2013'].astype('float')
dt12_streets_____________00_rn['2014'].astype('float')
dt12_streets_____________00_rn['2015'].astype('float')
dt12_streets_____________00_rn['2016'].astype('float')
dt12_streets_____________00_rn['2017'].astype('float')
dt12_streets_____________00_rn['2018'].astype('float')
dt12_streets_____________00_rn['2019'].astype('float')
dt12_streets_____________00_rn['2020'].astype('float')
dt12_streets_____________00_rn['2021'].astype('float')
dt12_streets_____________00_rn['2022'].astype('float')
dt12_streets_____________00_rn['2023'].astype('float')
```

[187]:
```
0      1.158776e+08
1      1.071491e+08
2      1.022051e+08
3      9.539000e+01
4      4.944024e+06
           …
913             NaN
914             NaN
915             NaN
916             NaN
917    0.000000e+00
Name: 2023, Length: 918, dtype: float64
```

[188]:
```
# 06.02.28.02
# change types
# dt13

'''dt13_ave_traffic_____________00_rn['2004_00'].astype('float')
dt13_ave_traffic_____________00_rn['2004_01'].astype('float')
dt13_ave_traffic_____________00_rn['2004_02'].astype('float')
```

```
dt13_ave_traffic_____________00_rn['2005_00'].astype('float')
dt13_ave_traffic_____________00_rn['2005_01'].astype('float')
dt13_ave_traffic_____________00_rn['2005_02'].astype('float')
dt13_ave_traffic_____________00_rn['2006_00'].astype('float')
dt13_ave_traffic_____________00_rn['2006_01'].astype('float')
dt13_ave_traffic_____________00_rn['2006_02'].astype('float')
dt13_ave_traffic_____________00_rn['2007_00'].astype('float')
dt13_ave_traffic_____________00_rn['2007_01'].astype('float')
dt13_ave_traffic_____________00_rn['2007_02'].astype('float')
dt13_ave_traffic_____________00_rn['2008_00'].astype('float')
dt13_ave_traffic_____________00_rn['2008_01'].astype('float')
dt13_ave_traffic_____________00_rn['2008_02'].astype('float')
dt13_ave_traffic_____________00_rn['2009_00'].astype('float')
dt13_ave_traffic_____________00_rn['2009_01'].astype('float')
dt13_ave_traffic_____________00_rn['2009_02'].astype('float')
dt13_ave_traffic_____________00_rn['2010_00'].astype('float')
dt13_ave_traffic_____________00_rn['2010_01'].astype('float')
dt13_ave_traffic_____________00_rn['2010_02'].astype('float')
dt13_ave_traffic_____________00_rn['2011_00'].astype('float')
dt13_ave_traffic_____________00_rn['2011_01'].astype('float')
dt13_ave_traffic_____________00_rn['2011_02'].astype('float')
dt13_ave_traffic_____________00_rn['2012_00'].astype('float')
dt13_ave_traffic_____________00_rn['2012_01'].astype('float')
dt13_ave_traffic_____________00_rn['2012_02'].astype('float')
dt13_ave_traffic_____________00_rn['2013_00'].astype('float')
dt13_ave_traffic_____________00_rn['2013_01'].astype('float')
dt13_ave_traffic_____________00_rn['2013_02'].astype('float')
dt13_ave_traffic_____________00_rn['2014_00'].astype('float')
dt13_ave_traffic_____________00_rn['2014_01'].astype('float')
dt13_ave_traffic_____________00_rn['2014_02'].astype('float')
dt13_ave_traffic_____________00_rn['2015_00'].astype('float')
dt13_ave_traffic_____________00_rn['2015_01'].astype('float')
dt13_ave_traffic_____________00_rn['2015_02'].astype('float')
dt13_ave_traffic_____________00_rn['2016_00'].astype('float')
dt13_ave_traffic_____________00_rn['2016_01'].astype('float')
dt13_ave_traffic_____________00_rn['2016_02'].astype('float')
dt13_ave_traffic_____________00_rn['2017_00'].astype('float')
dt13_ave_traffic_____________00_rn['2017_01'].astype('float')
dt13_ave_traffic_____________00_rn['2017_02'].astype('float')
dt13_ave_traffic_____________00_rn['2018_00'].astype('float')
dt13_ave_traffic_____________00_rn['2018_01'].astype('float')
dt13_ave_traffic_____________00_rn['2018_02'].astype('float')
dt13_ave_traffic_____________00_rn['2019_00'].astype('float')
dt13_ave_traffic_____________00_rn['2019_01'].astype('float')
dt13_ave_traffic_____________00_rn['2019_02'].astype('float')
dt13_ave_traffic_____________00_rn['2020_00'].astype('float')
dt13_ave_traffic_____________00_rn['2020_01'].astype('float')
```

```
dt13_ave_traffic_____________00_rn['2020_02'].astype('float')
dt13_ave_traffic_____________00_rn['2021_00'].astype('float')
dt13_ave_traffic_____________00_rn['2021_01'].astype('float')
dt13_ave_traffic_____________00_rn['2021_02'].astype('float')
dt13_ave_traffic_____________00_rn['2022_00'].astype('float')
dt13_ave_traffic_____________00_rn['2022_01'].astype('float')
dt13_ave_traffic_____________00_rn['2022_02'].astype('float')
dt13_ave_traffic_____________00_rn['2023_00'].astype('float')
dt13_ave_traffic_____________00_rn['2023_01'].astype('float')
dt13_ave_traffic_____________00_rn['2023_02'].astype('float')'''
```

[188]:
```
"dt13_ave_traffic_____________00_rn['2004_00'].astype('float')\ndt13_ave_traffic
_____________00_rn['2004_01'].astype('float')\ndt13_ave_traffic_____________00_r
n['2004_02'].astype('float')\ndt13_ave_traffic_____________00_rn['2005_00'].asty
pe('float')\ndt13_ave_traffic_____________00_rn['2005_01'].astype('float')\ndt13
_ave_traffic_____________00_rn['2005_02'].astype('float')\ndt13_ave_traffic_____
_____00_rn['2006_00'].astype('float')\ndt13_ave_traffic_____________00_rn['20
06_01'].astype('float')\ndt13_ave_traffic_____________00_rn['2006_02'].astype('f
loat')\ndt13_ave_traffic_____________00_rn['2007_00'].astype('float')\ndt13_ave_
traffic_____________00_rn['2007_01'].astype('float')\ndt13_ave_traffic_________
___00_rn['2007_02'].astype('float')\ndt13_ave_traffic_____________00_rn['2008_00
'].astype('float')\ndt13_ave_traffic_____________00_rn['2008_01'].astype('float'
)\ndt13_ave_traffic_____________00_rn['2008_02'].astype('float')\ndt13_ave_traff
ic_____________00_rn['2009_00'].astype('float')\ndt13_ave_traffic_____________00
_rn['2009_01'].astype('float')\ndt13_ave_traffic_____________00_rn['2009_02'].as
type('float')\ndt13_ave_traffic_____________00_rn['2010_00'].astype('float')\ndt
13_ave_traffic_____________00_rn['2010_01'].astype('float')\ndt13_ave_traffic___
_________00_rn['2010_02'].astype('float')\ndt13_ave_traffic_____________00_rn['
2011_00'].astype('float')\ndt13_ave_traffic_____________00_rn['2011_01'].astype(
'float')\ndt13_ave_traffic_____________00_rn['2011_02'].astype('float')\ndt13_av
e_traffic_____________00_rn['2012_00'].astype('float')\ndt13_ave_traffic________
_____00_rn['2012_01'].astype('float')\ndt13_ave_traffic_____________00_rn['2012_
02'].astype('float')\ndt13_ave_traffic_____________00_rn['2013_00'].astype('floa
t')\ndt13_ave_traffic_____________00_rn['2013_01'].astype('float')\ndt13_ave_tra
ffic_____________00_rn['2013_02'].astype('float')\ndt13_ave_traffic_____________
00_rn['2014_00'].astype('float')\ndt13_ave_traffic_____________00_rn['2014_01'].
astype('float')\ndt13_ave_traffic_____________00_rn['2014_02'].astype('float')\n
dt13_ave_traffic_____________00_rn['2015_00'].astype('float')\ndt13_ave_traffic_
____________00_rn['2015_01'].astype('float')\ndt13_ave_traffic_____________00_rn
['2015_02'].astype('float')\ndt13_ave_traffic_____________00_rn['2016_00'].astyp
e('float')\ndt13_ave_traffic_____________00_rn['2016_01'].astype('float')\ndt13_
ave_traffic_____________00_rn['2016_02'].astype('float')\ndt13_ave_traffic_____
_______00_rn['2017_00'].astype('float')\ndt13_ave_traffic_____________00_rn['201
7_01'].astype('float')\ndt13_ave_traffic_____________00_rn['2017_02'].astype('fl
oat')\ndt13_ave_traffic_____________00_rn['2018_00'].astype('float')\ndt13_ave_t
raffic_____________00_rn['2018_01'].astype('float')\ndt13_ave_traffic_________
__00_rn['2018_02'].astype('float')\ndt13_ave_traffic_____________00_rn['2019_00'
```

```
].astype('float')\ndt13_ave_traffic_____________00_rn['2019_01'].astype('float')
\ndt13_ave_traffic_____________00_rn['2019_02'].astype('float')\ndt13_ave_traffi
c_____________00_rn['2020_00'].astype('float')\ndt13_ave_traffic_____________00_
rn['2020_01'].astype('float')\ndt13_ave_traffic_____________00_rn['2020_02'].ast
ype('float')\ndt13_ave_traffic_____________00_rn['2021_00'].astype('float')\ndt1
3_ave_traffic_____________00_rn['2021_01'].astype('float')\ndt13_ave_traffic____
_________00_rn['2021_02'].astype('float')\ndt13_ave_traffic_____________00_rn['2
022_00'].astype('float')\ndt13_ave_traffic_____________00_rn['2022_01'].astype('
float')\ndt13_ave_traffic_____________00_rn['2022_02'].astype('float')\ndt13_ave
_traffic_____________00_rn['2023_00'].astype('float')\ndt13_ave_traffic_________
____00_rn['2023_01'].astype('float')\ndt13_ave_traffic_____________00_rn['2023_0
2'].astype('float')"
```

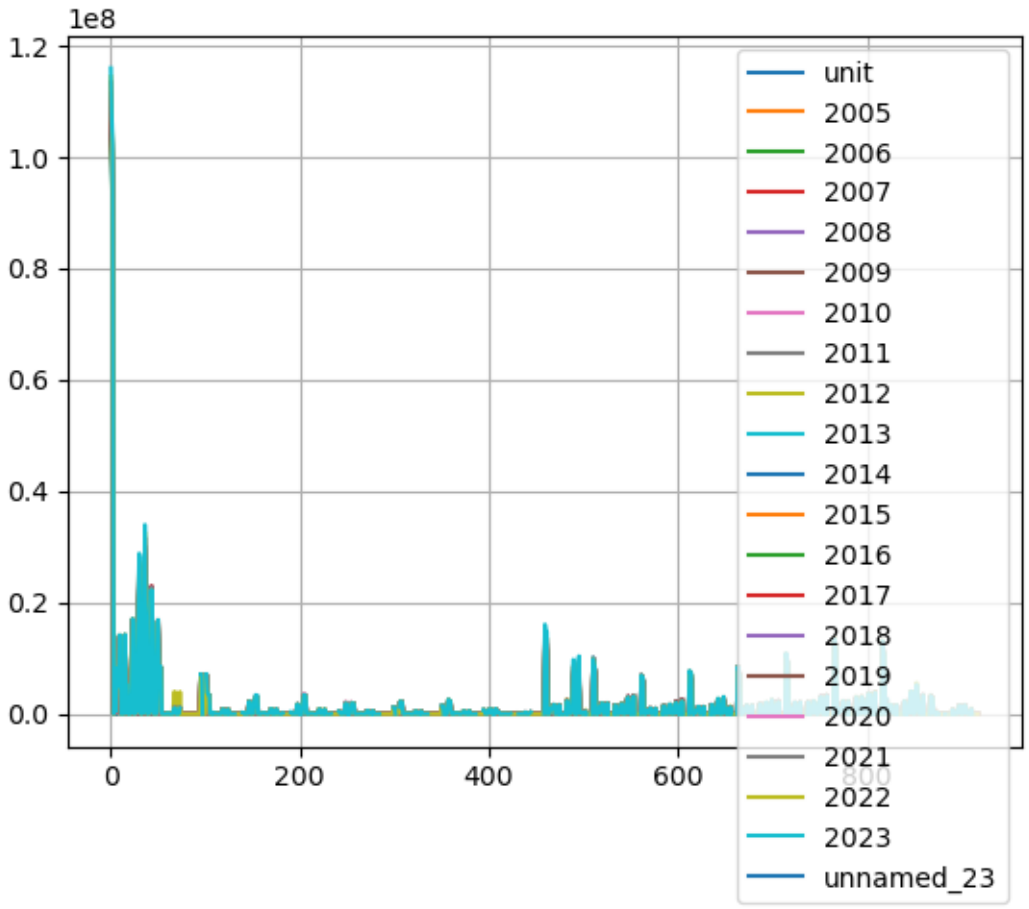
[189]:
```
# 06.02.28.03
# change types
# dt14

'''dt14_yearly_____________________00_rn['2003_00'].astype('float')
dt14_yearly_____________________00_rn['2003_01'].astype('float')
dt14_yearly_____________________00_rn['2004_00'].astype('float')
dt14_yearly_____________________00_rn['2004_01'].astype('float')
dt14_yearly_____________________00_rn['2005_00'].astype('float')
dt14_yearly_____________________00_rn['2005_01'].astype('float')
dt14_yearly_____________________00_rn['2006_00'].astype('float')
dt14_yearly_____________________00_rn['2006_01'].astype('float')
dt14_yearly_____________________00_rn['2007_00'].astype('float')
dt14_yearly_____________________00_rn['2007_01'].astype('float')
dt14_yearly_____________________00_rn['2008_00'].astype('float')
dt14_yearly_____________________00_rn['2008_01'].astype('float')
dt14_yearly_____________________00_rn['2009_00'].astype('float')
dt14_yearly_____________________00_rn['2009_01'].astype('float')
dt14_yearly_____________________00_rn['2010_00'].astype('float')
dt14_yearly_____________________00_rn['2010_01'].astype('float')
dt14_yearly_____________________00_rn['2011_00'].astype('float')
dt14_yearly_____________________00_rn['2011_01'].astype('float')
dt14_yearly_____________________00_rn['2012_00'].astype('float')
dt14_yearly_____________________00_rn['2012_01'].astype('float')
dt14_yearly_____________________00_rn['2013_00'].astype('float')
dt14_yearly_____________________00_rn['2013_01'].astype('float')
dt14_yearly_____________________00_rn['2014_00'].astype('float')
dt14_yearly_____________________00_rn['2014_01'].astype('float')
dt14_yearly_____________________00_rn['2015_00'].astype('float')
dt14_yearly_____________________00_rn['2015_01'].astype('float')
dt14_yearly_____________________00_rn['2016_00'].astype('float')
dt14_yearly_____________________00_rn['2016_01'].astype('float')
dt14_yearly_____________________00_rn['2017_00'].astype('float')
dt14_yearly_____________________00_rn['2017_01'].astype('float')
```

```
dt14_yearly_____________00_rn['2018_00'].astype('float')
dt14_yearly_____________00_rn['2018_01'].astype('float')
dt14_yearly_____________00_rn['2019_00'].astype('float')
dt14_yearly_____________00_rn['2019_01'].astype('float')
dt14_yearly_____________00_rn['2020_00'].astype('float')
dt14_yearly_____________00_rn['2020_01'].astype('float')
dt14_yearly_____________00_rn['2021_00'].astype('float')
dt14_yearly_____________00_rn['2021_01'].astype('float')
dt14_yearly_____________00_rn['2022_00'].astype('float')
dt14_yearly_____________00_rn['2022_01'].astype('float')
dt14_yearly_____________00_rn['2023_00'].astype('float')
dt14_yearly_____________00_rn['2023_01'].astype('float')'''
```

[189]:
```
"dt14_yearly_____________00_rn['2003_00'].astype('float')\ndt14_yearly_____
_____________00_rn['2003_01'].astype('float')\ndt14_yearly_____________00_r
n['2004_00'].astype('float')\ndt14_yearly_____________00_rn['2004_01'].asty
pe('float')\ndt14_yearly_____________00_rn['2005_00'].astype('float')\ndt14
_yearly_____________00_rn['2005_01'].astype('float')\ndt14_yearly_________
_____00_rn['2006_00'].astype('float')\ndt14_yearly_____________00_rn['20
06_01'].astype('float')\ndt14_yearly_____________00_rn['2007_00'].astype('f
loat')\ndt14_yearly_____________00_rn['2007_01'].astype('float')\ndt14_year
ly_____________00_rn['2008_00'].astype('float')\ndt14_yearly_____________
___00_rn['2008_01'].astype('float')\ndt14_yearly_____________00_rn['2009_00
'].astype('float')\ndt14_yearly_____________00_rn['2009_01'].astype('float'
)\ndt14_yearly_____________00_rn['2010_00'].astype('float')\ndt14_yearly___
_____________00_rn['2010_01'].astype('float')\ndt14_yearly_____________00
_rn['2011_00'].astype('float')\ndt14_yearly_____________00_rn['2011_01'].as
type('float')\ndt14_yearly_____________00_rn['2012_00'].astype('float')\ndt
14_yearly_____________00_rn['2012_01'].astype('float')\ndt14_yearly_______
_____00_rn['2013_00'].astype('float')\ndt14_yearly_____________00_rn['
2013_01'].astype('float')\ndt14_yearly_____________00_rn['2014_00'].astype(
'float')\ndt14_yearly_____________00_rn['2014_01'].astype('float')\ndt14_ye
arly_____________00_rn['2015_00'].astype('float')\ndt14_yearly_____________
_____00_rn['2015_01'].astype('float')\ndt14_yearly_____________00_rn['2016_
00'].astype('float')\ndt14_yearly_____________00_rn['2016_01'].astype('floa
t')\ndt14_yearly_____________00_rn['2017_00'].astype('float')\ndt14_yearly_
_____________00_rn['2017_01'].astype('float')\ndt14_yearly_____________
00_rn['2018_00'].astype('float')\ndt14_yearly_____________00_rn['2018_01'].
astype('float')\ndt14_yearly_____________00_rn['2019_00'].astype('float')\n
dt14_yearly_____________00_rn['2019_01'].astype('float')\ndt14_yearly_____
_____________00_rn['2020_00'].astype('float')\ndt14_yearly_____________00_rn
['2020_01'].astype('float')\ndt14_yearly_____________00_rn['2021_00'].astyp
e('float')\ndt14_yearly_____________00_rn['2021_01'].astype('float')\ndt14_
yearly_____________00_rn['2022_00'].astype('float')\ndt14_yearly_________
_____00_rn['2022_01'].astype('float')\ndt14_yearly_____________00_rn['202
3_00'].astype('float')\ndt14_yearly_____________00_rn['2023_01'].astype('fl
oat')"
```

[190]:
```
# 06.02.29.01
# return basic plot
# rendered basic plot to see visually and to determine if data is numeric
# dt12

plt.figure(figsize=(18,8))
dt12_streets_____________00_rn.plot()
plt.box(True)
plt.grid(True)
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```



[191]:
```
# 06.02.29.02
# return basic plot
```

```
# rendered basic plot to see visually and to determine if data is numeric
# dt13

'''plt.figure(figsize=(18,8))
dt13_ave_traffic_____________00_rn.plot()
plt.box(True)
plt.grid(True)
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()'''
```

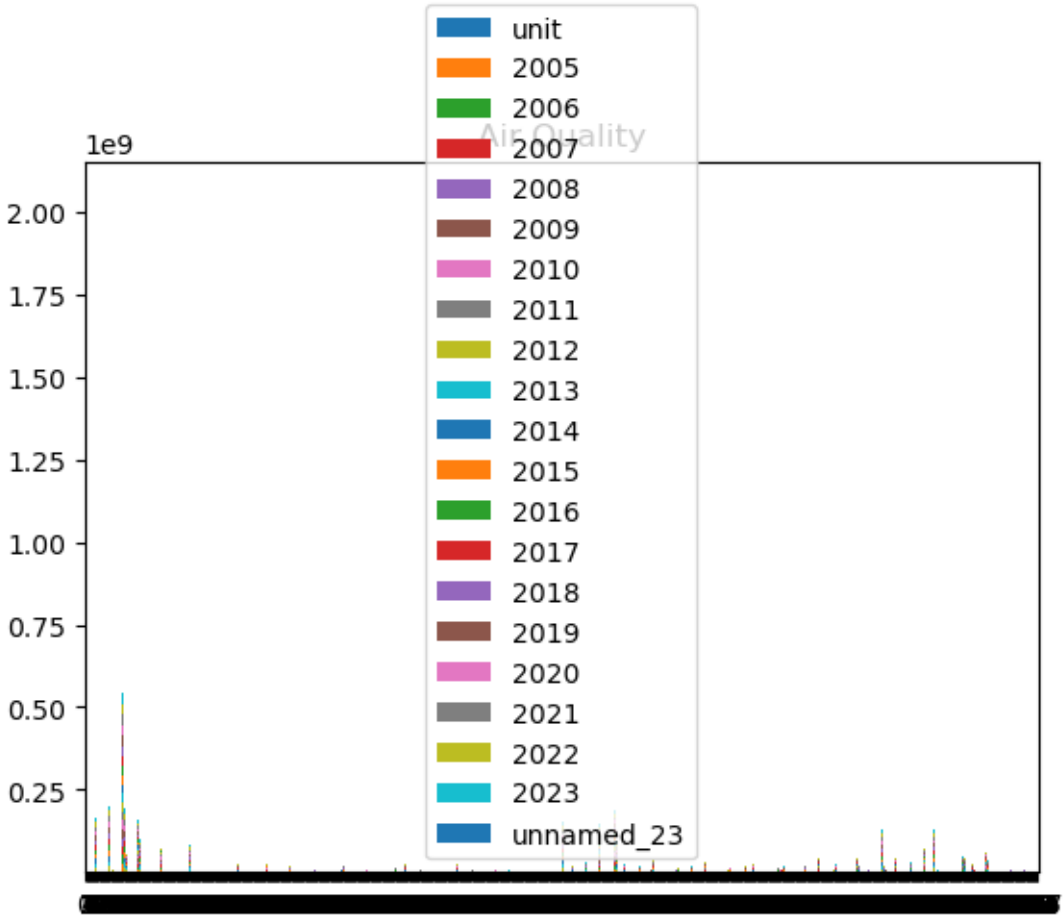
[191]:
```
"plt.figure(figsize=(18,8))\ndt13_ave_traffic_____________00_rn.plot()\nplt.box(
True)\nplt.grid(True)\nplt.title('', fontsize = 16, color =
'#0047ab')\nplt.xlabel('', fontsize = 14, color = '#0047ab')\nplt.ylabel('',
fontsize = 14, color = '#0047ab')\nplt.show()"
```

[192]:
```
# 06.02.29.03
# return basic plot
# rendered basic plot to see visually and to determine if data is numeric
# dt14

'''plt.figure(figsize=(18,8))
dt14_yearly_________________00_rn.plot()
plt.box(True)
plt.grid(True)
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()'''
```

[192]:
```
"plt.figure(figsize=(18,8))\ndt14_yearly_________________00_rn.plot()\nplt.box(
True)\nplt.grid(True)\nplt.title('', fontsize = 16, color =
'#0047ab')\nplt.xlabel('', fontsize = 14, color = '#0047ab')\nplt.ylabel('',
fontsize = 14, color = '#0047ab')\nplt.show()"
```

[193]:
```
# 06.02.30.01
# render bar chart
# dt12

plt.figure(figsize=(18,8))
dt12_streets_________________00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```



[194]:
```
# 06.02.30.02
# render bar chart
# dt13

'''plt.figure(figsize=(18,8))
dt13_ave_traffic_____________00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()'''
```

[194]:
```
"plt.figure(figsize=(18,8))\ndt13_ave_traffic_____________00_rn.plot(kind='bar',
stacked=True)\nplt.title('Air Quality')\nplt.xticks(rotation=0,
ha='center')\nplt.show()"
```

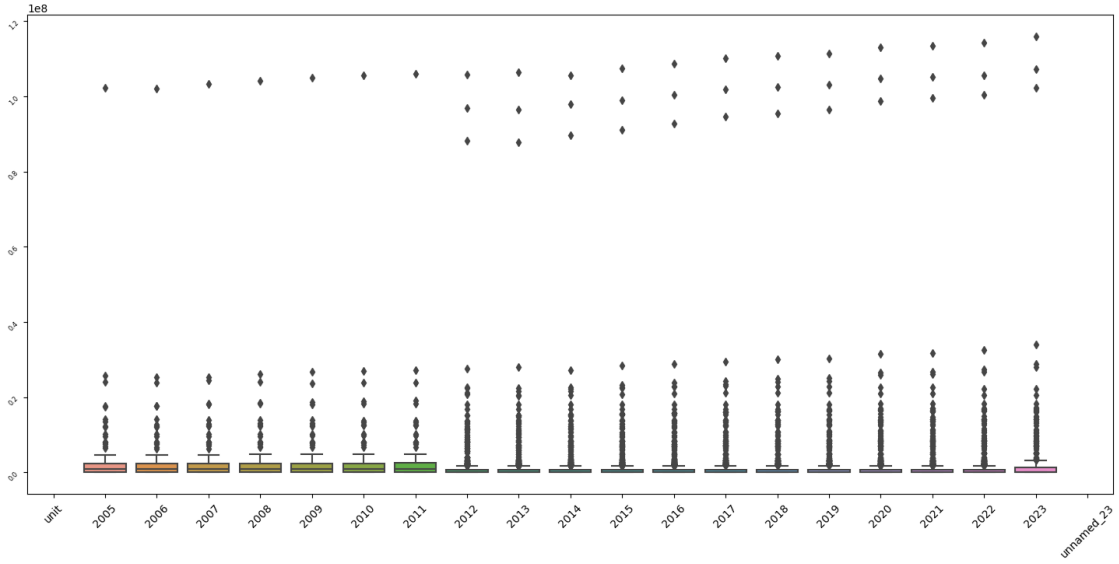
[195]:
```
# 06.02.30.03
# render bar chart
# dt14
```

```
'''plt.figure(figsize=(18,8))
dt14_yearly__________________00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()'''
```

[195]:
```
"plt.figure(figsize=(18,8))\ndt14_yearly__________________00_rn.plot(kind='bar',
stacked=True)\nplt.title('Air Quality')\nplt.xticks(rotation=0,
ha='center')\nplt.show()"
```

[196]:
```
# 06.02.31.01
# return columns
# dt12

plt.figure(figsize=(18,8))
sns.boxplot(dt12_streets_____________00_rn)
plt.xticks(fontsize = 10)
plt.xticks(rotation = 45)
plt.yticks(fontsize = 6)
plt.yticks(rotation = 45)
```

[196]:
```
(array([-2.0e+07,  0.0e+00,  2.0e+07,  4.0e+07,  6.0e+07,  8.0e+07,
         1.0e+08,  1.2e+08,  1.4e+08]),
 [Text(0, -20000000.0, '-0.2'),
  Text(0, 0.0, '0.0'),
  Text(0, 20000000.0, '0.2'),
  Text(0, 40000000.0, '0.4'),
  Text(0, 60000000.0, '0.6'),
  Text(0, 80000000.0, '0.8'),
  Text(0, 100000000.0, '1.0'),
  Text(0, 120000000.0, '1.2'),
  Text(0, 140000000.0, '1.4')])
```



[197]:
```
# 06.02.31.02
# return columns
# dt13

'''plt.figure(figsize=(18,8))
sns.boxplot(dt13_ave_traffic_____________00_rn)
plt.xticks(fontsize = 10)
plt.xticks(rotation = 45)
plt.yticks(fontsize = 6)
plt.yticks(rotation = 45)'''
```

[197]:
```
'plt.figure(figsize=(18,8))\nsns.boxplot(dt13_ave_traffic_____________00_rn)\npl
t.xticks(fontsize = 10)\nplt.xticks(rotation = 45)\nplt.yticks(fontsize =
6)\nplt.yticks(rotation = 45)'
```

[198]:
```
# 06.02.31.03
# return columns
# dt14

'''plt.figure(figsize=(18,8))
sns.boxplot(dt14_yearly_____________________00_rn)
plt.xticks(fontsize = 10)
plt.xticks(rotation = 45)
plt.yticks(fontsize = 6)
plt.yticks(rotation = 45)'''
```

[198]:
```
'plt.figure(figsize=(18,8))\nsns.boxplot(dt14_yearly_____________________00_rn)\npl
t.xticks(fontsize = 10)\nplt.xticks(rotation = 45)\nplt.yticks(fontsize =
```

```
6)\nplt.yticks(rotation = 45)'
```

[199]:
```
# 06.02.32.01
# read csv
# assign variable
# dt15

dt15_spatial_____________________00 = pd.read_csv('input_data_spatial_panel.csv')
```

[200]:
```
# 06.02.32.02
# return first and last ten rows
# dt015

print(dt15_spatial_____________________00.head(10))
print(dt15_spatial_____________________00.tail(10))
```

```
   Unnamed: 0  k_so2  k_no2  k_pm25  …  NNW  OTH_WD  k_so2_lag  k_no2_lag
0           1  0.005  0.068    49.0  …    1       0      0.004      0.067
1           2  0.003  0.021    20.0  …    0       1      0.005      0.068
2           3  0.006  0.067    35.0  …    0       0      0.003      0.021
3           4  0.004  0.061    26.0  …    1       0      0.006      0.067
4           5  0.004  0.059    28.0  …    0       1      0.004      0.061
5           6  0.007  0.031    64.0  …    0       1      0.004      0.059
6           7  0.006  0.056    71.0  …    0       0      0.007      0.031
7           8  0.003  0.046    80.0  …    0       0      0.006      0.056
8           9  0.004  0.048    70.0  …    0       0      0.003      0.046
9          10  0.005  0.024    58.0  …    0       0      0.004      0.048

[10 rows x 19 columns]
       Unnamed: 0     k_so2     k_no2  …  OTH_WD  k_so2_lag  k_no2_lag
28106       28502  0.004500  0.006200  …       0   0.004100   0.005300
28107       28503  0.002200  0.005200  …       0   0.004500   0.006200
28108       28504  0.002200  0.004300  …       0   0.002200   0.005200
28109       28505  0.002707  0.007261  …       0   0.002200   0.004300
28110       28506  0.002707  0.007261  …       0   0.002707   0.007261
28111       28507  0.001000  0.006500  …       0   0.002707   0.007261
28112       28508  0.000800  0.009200  …       0   0.001000   0.006500
28113       28509  0.000900  0.003200  …       0   0.000800   0.009200
28114       28510  0.001000  0.002700  …       0   0.000900   0.003200
28115       28511  0.001200  0.002400  …       0   0.001000   0.002700

[10 rows x 19 columns]
```

[201]:
```
# 06.02.32.03
# return dimensions
# dt15

print(dt15_spatial_____________________00.shape)
```

```
(28116, 19)
```

[202]:
```
# 06.02.32.04
# confirm column names
# dt15

print(dt15_spatial_____________00.columns)
```

```
Index(['Unnamed: 0', 'k_so2', 'k_no2', 'k_pm25', 'temp', 'rain', 'ws', 'wd',
       'humidity', 'pressure', 'c_pm25_mean', 'id2', 'date_id', 'WSW', 'WNW',
       'NNW', 'OTH_WD', 'k_so2_lag', 'k_no2_lag'],
      dtype='object')
```

[203]:
```
# 06.02.32.05
# column rename to remove spaces and quotes
# dt15

dt15_spatial_____________00_rn = dt15_spatial_____________00.
 ↪rename(columns = {
    'Unnamed: 0': 'unnamed',
    'k_so2': 'k_so2',
    'k_no2': 'k_no2',
    'k_pm25': 'k_pm25',
    'temp': 'temp',
    'rain': 'rain',
    'ws': 'ws',
    'wd': 'wd',
    'humidity': 'humidity',
    'pressure': 'pressure',
    'c_pm25_mean': 'c_pm25_mean',
    'id2': 'id2',
    'date_id': 'date_id',
    'WSW': 'WSW',
    'WNW': 'WNW',
    'NNW': 'NNW',
    'OTH_WD': 'OTH_WD',
    'k_so2_lag': 'k_so2_lag',
    'k_no2_lag': 'k_no2_lag'
    })
```

[204]:
```
# 06.02.32.06
# confirm column names
# dt15

print(dt15_spatial_____________00_rn.columns)
```

```
Index(['unnamed', 'k_so2', 'k_no2', 'k_pm25', 'temp', 'rain', 'ws', 'wd',
       'humidity', 'pressure', 'c_pm25_mean', 'id2', 'date_id', 'WSW', 'WNW',
       'NNW', 'OTH_WD', 'k_so2_lag', 'k_no2_lag'],
```

```
      dtype='object')
```

[205]:
```
# 06.02.32.07
# confirm types
# dt15

print(dt15_spatial_____________00_rn['unnamed'].dtypes)
print(dt15_spatial_____________00_rn['k_so2'].dtypes)
print(dt15_spatial_____________00_rn['k_no2'].dtypes)
print(dt15_spatial_____________00_rn['k_pm25'].dtypes)
print(dt15_spatial_____________00_rn['temp'].dtypes)
print(dt15_spatial_____________00_rn['rain'].dtypes)
print(dt15_spatial_____________00_rn['ws'].dtypes)
print(dt15_spatial_____________00_rn['wd'].dtypes)
print(dt15_spatial_____________00_rn['humidity'].dtypes)
print(dt15_spatial_____________00_rn['pressure'].dtypes)
print(dt15_spatial_____________00_rn['c_pm25_mean'].dtypes)
print(dt15_spatial_____________00_rn['id2'].dtypes)
print(dt15_spatial_____________00_rn['date_id'].dtypes)
print(dt15_spatial_____________00_rn['WSW'].dtypes)
print(dt15_spatial_____________00_rn['WNW'].dtypes)
print(dt15_spatial_____________00_rn['NNW'].dtypes)
print(dt15_spatial_____________00_rn['OTH_WD'].dtypes)
print(dt15_spatial_____________00_rn['k_so2_lag'].dtypes)
print(dt15_spatial_____________00_rn['k_no2_lag'].dtypes)
```

```
int64
float64
float64
float64
float64
float64
float64
float64
float64
float64
float64
int64
int64
int64
int64
int64
int64
float64
float64
```

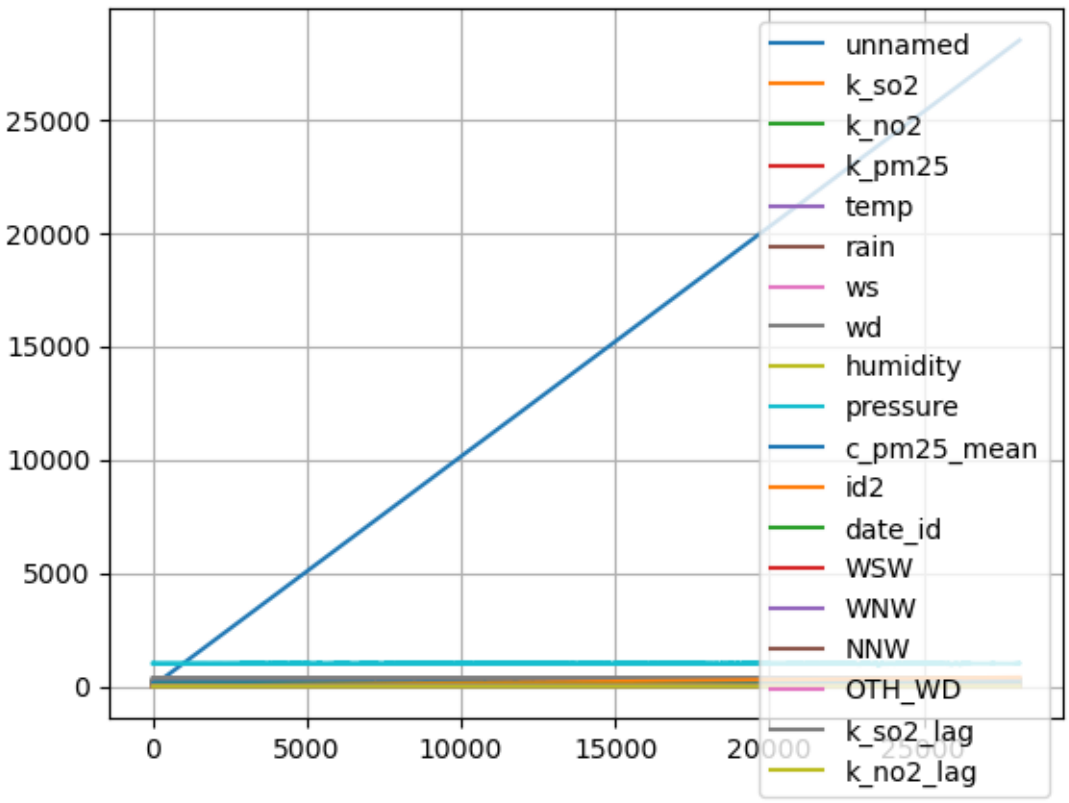
[206]:
```
# 06.02.32.08
# change types
```

```
# dt15

'''dt15_spatial_____________00_rn['2005'].astype('float')
dt15_spatial_____________00_rn['2006'].astype('float')
dt15_spatial_____________00_rn['2007'].astype('float')'''
```

[206]:
```
"dt15_spatial_____________00_rn['2005'].astype('float')\ndt15_spatial_____
_____00_rn['2006'].astype('float')\ndt15_spatial_____________00_rn['200
7'].astype('float')"
```

[207]:
```
# 06.02.32.09
# return basic plot
# rendered basic plot to see visually and to determine if data is numeric
# dt15

plt.figure(figsize=(18,8))
dt15_spatial_____________00_rn.plot()
plt.box(True)
plt.grid(True)
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```



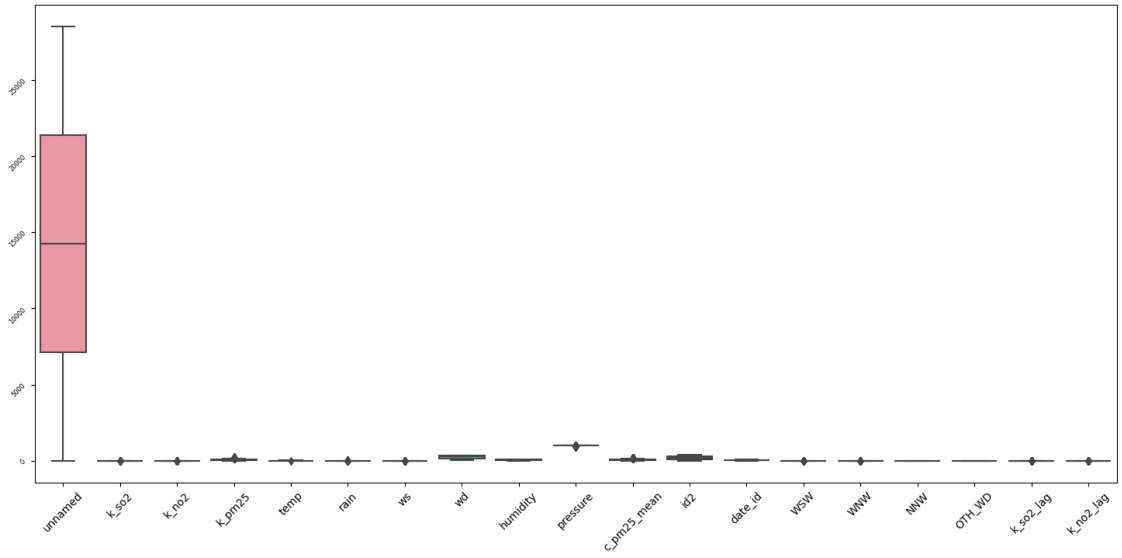
[208]:
```
# 06.02.32.10
# render bar chart
# dt15

'''plt.figure(figsize=(18,8))
dt15_spatial_____________________00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()'''
```

[208]:
```
"plt.figure(figsize=(18,8))\ndt15_spatial_____________________00_rn.plot(kind='bar',
stacked=True)\nplt.title('Air Quality')\nplt.xticks(rotation=0,
ha='center')\nplt.show()"
```

[209]:
```
# 06.02.32.11
# return columns
# dt15

plt.figure(figsize=(18,8))
sns.boxplot(dt15_spatial_____________________00_rn)
plt.xticks(fontsize = 10)
```

```
plt.xticks(rotation = 45)
plt.yticks(fontsize = 6)
plt.yticks(rotation = 45)
```

[209]:
```
(array([-5000.,     0.,  5000., 10000., 15000., 20000., 25000., 30000.]),
 [Text(0, -5000.0, '-5000'),
  Text(0, 0.0, '0'),
  Text(0, 5000.0, '5000'),
  Text(0, 10000.0, '10000'),
  Text(0, 15000.0, '15000'),
  Text(0, 20000.0, '20000'),
  Text(0, 25000.0, '25000'),
  Text(0, 30000.0, '30000')])
```



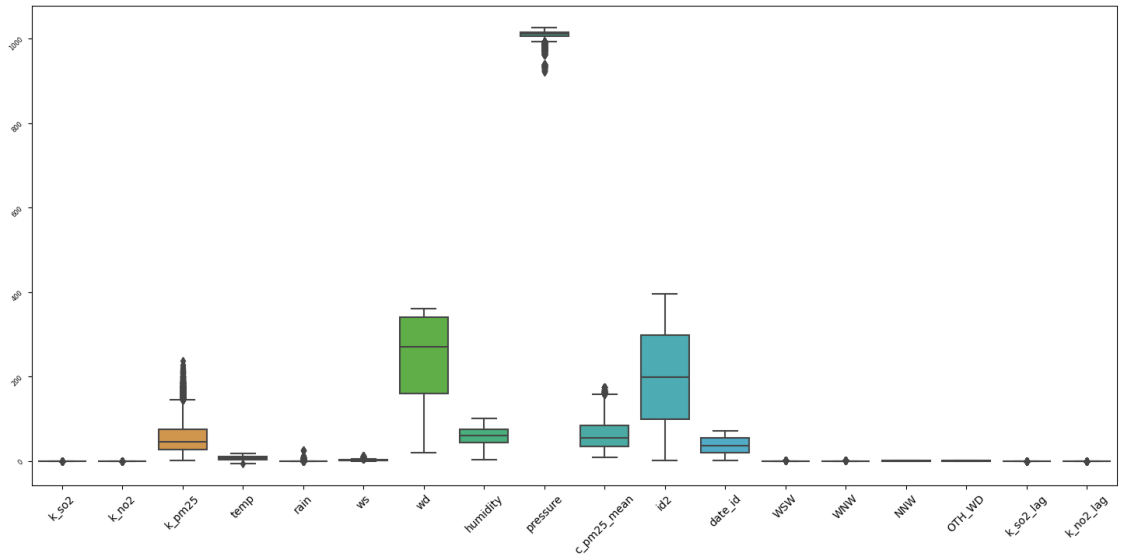
[210]:
```
# 06.02.32.12
# drop unnamed column
# dt15

dt15_spatial_____________00_rn_drp = dt15_spatial_____________00_rn.
  ↪drop(['unnamed'], axis=1)
```

[213]:
```
# 06.02.32.13
# replace NA with mode
# dt04

dt15_spatial_____________00_rn_drp['k_so2']=dt15_spatial_____________00_rn_drp['k_so2'
  ↪fillna(dt15_spatial_____________00_rn_drp['k_so2'].mode()[0])
```

```
dt15_spatial_____________00_rn_drp['k_no2']=dt15_spatial_____________00_rn_drp['k_no2'
 ↪fillna(dt15_spatial_____________00_rn_drp['k_no2'].mode()[0])
dt15_spatial_____________00_rn_drp['k_pm25']=dt15_spatial_____________00_rn_drp['k_pm2
 ↪fillna(dt15_spatial_____________00_rn_drp['k_pm25'].mode()[0])
dt15_spatial_____________00_rn_drp['temp']=dt15_spatial_____________00_rn_drp['temp'].
 ↪fillna(dt15_spatial_____________00_rn_drp['temp'].mode()[0])
dt15_spatial_____________00_rn_drp['rain']=dt15_spatial_____________00_rn_drp['rain'].
 ↪fillna(dt15_spatial_____________00_rn_drp['rain'].mode()[0])
dt15_spatial_____________00_rn_drp['ws']=dt15_spatial_____________00_rn_drp['ws'].
 ↪fillna(dt15_spatial_____________00_rn_drp['ws'].mode()[0])
dt15_spatial_____________00_rn_drp['wd']=dt15_spatial_____________00_rn_drp['wd'].
 ↪fillna(dt15_spatial_____________00_rn_drp['wd'].mode()[0])
dt15_spatial_____________00_rn_drp['humidity']=dt15_spatial_____________00_rn_drp['hum
 ↪fillna(dt15_spatial_____________00_rn_drp['humidity'].mode()[0])
dt15_spatial_____________00_rn_drp['pressure']=dt15_spatial_____________00_rn_drp['pre
 ↪fillna(dt15_spatial_____________00_rn_drp['pressure'].mode()[0])
dt15_spatial_____________00_rn_drp['c_pm25_mean']=dt15_spatial_____________00_rn_drp['
 ↪fillna(dt15_spatial_____________00_rn_drp['c_pm25_mean'].mode()[0])
dt15_spatial_____________00_rn_drp['id2']=dt15_spatial_____________00_rn_drp['id2'].
 ↪fillna(dt15_spatial_____________00_rn_drp['id2'].mode()[0])
dt15_spatial_____________00_rn_drp['date_id']=dt15_spatial_____________00_rn_drp['date
 ↪fillna(dt15_spatial_____________00_rn_drp['date_id'].mode()[0])
dt15_spatial_____________00_rn_drp['WSW']=dt15_spatial_____________00_rn_drp['WSW'].
 ↪fillna(dt15_spatial_____________00_rn_drp['WSW'].mode()[0])
dt15_spatial_____________00_rn_drp['WNW']=dt15_spatial_____________00_rn_drp['WNW'].
 ↪fillna(dt15_spatial_____________00_rn_drp['WNW'].mode()[0])
dt15_spatial_____________00_rn_drp['NNW']=dt15_spatial_____________00_rn_drp['NNW'].
 ↪fillna(dt15_spatial_____________00_rn_drp['NNW'].mode()[0])
dt15_spatial_____________00_rn_drp['OTH_WD']=dt15_spatial_____________00_rn_drp['OTH_W
 ↪fillna(dt15_spatial_____________00_rn_drp['OTH_WD'].mode()[0])
dt15_spatial_____________00_rn_drp['k_so2_lag']=dt15_spatial_____________00_rn_drp['k_
 ↪fillna(dt15_spatial_____________00_rn_drp['k_so2_lag'].mode()[0])
dt15_spatial_____________00_rn_drp['k_no2_lag']=dt15_spatial_____________00_rn_drp['k_
 ↪fillna(dt15_spatial_____________00_rn_drp['k_no2_lag'].mode()[0])
```

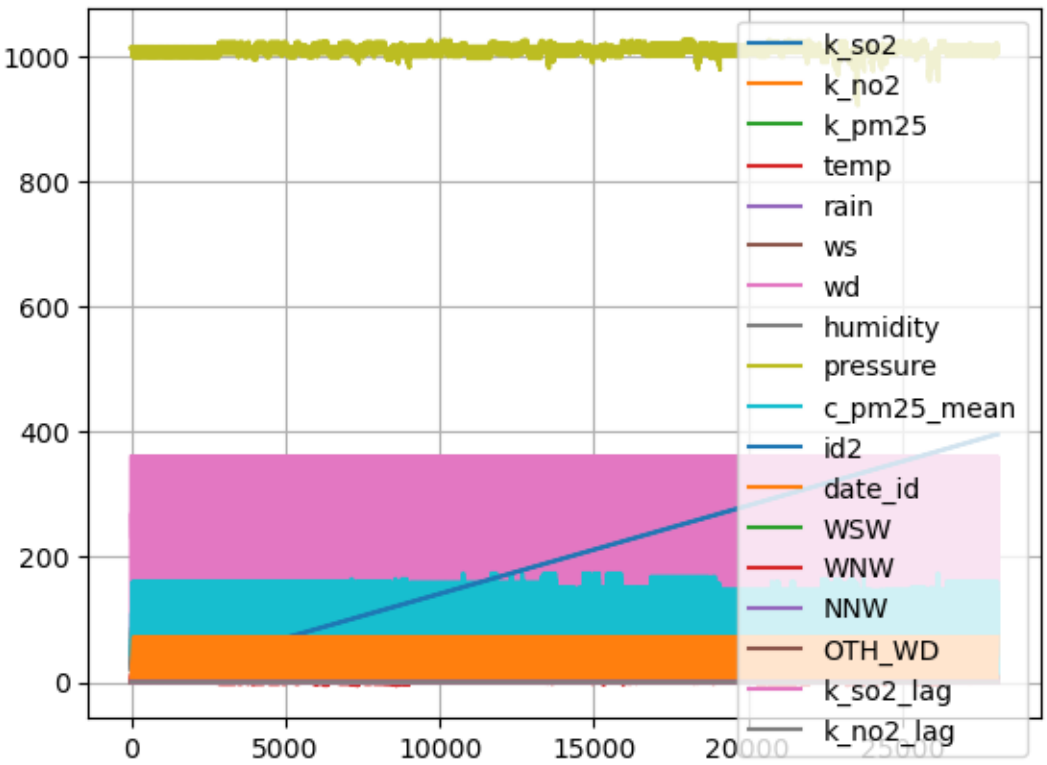
[214]:
```
# 06.02.32.14
# return columns
# dt15

plt.figure(figsize=(18,8))
sns.boxplot(dt15_spatial_____________00_rn_drp)
plt.xticks(fontsize = 10)
plt.xticks(rotation = 45)
plt.yticks(fontsize = 6)
plt.yticks(rotation = 45)
```

[214]:
```
(array([-200.,    0.,  200.,  400.,  600.,  800., 1000., 1200.]),
 [Text(0, -200.0, '-200'),
  Text(0, 0.0, '0'),
  Text(0, 200.0, '200'),
  Text(0, 400.0, '400'),
  Text(0, 600.0, '600'),
  Text(0, 800.0, '800'),
  Text(0, 1000.0, '1000'),
  Text(0, 1200.0, '1200')])
```



[215]:
```
# 06.02.32.15
# return basic plot
# rendered basic plot to see visually and to determine if data is numeric
# dt15

plt.figure(figsize=(18,8))
dt15_spatial_____________00_rn_drp.plot()
plt.box(True)
plt.grid(True)
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```



[216]:
```
# 06.02.33.01
# preparing data for modeling
# add index column
# index column to select specific rows
# dt15

dt15_spatial_____________________00_rn_drp.insert(0, 'index', range(0, 0 +␣
 ↪len(dt15_spatial_____________________00_rn_drp)))
```

[217]:
```
# 06.02.33.02
# preparing data for modeling
# create dummy variables
# due to returning boolean values, converting dummmies to integers
# dt15

dt15_spatial_____________________00_rn_drp_dv = pd.
 ↪get_dummies(dt15_spatial_____________________00_rn_drp, drop_first = True, dtype␣
 ↪= int)
```

[218]:
```
# 06.02.33.03
# preparing data for modeling
# split data
```

```
# select columns
# dt15

dt15_x01 = dt15_spatial_____________00_rn_drp_dv.drop(['id2'], axis = 1)
dt15_y01 = dt15_spatial_____________00_rn_drp_dv['id2']
```

[219]:
```
# 06.02.33.04
# preparing data for modeling
# split into train and test
# dt15

dt15_x01_trn, dt15_x01_tst, dt15_y01_trn, dt15_y01_tst =␣
 ↪train_test_split(dt15_x01, dt15_y01, test_size = 0.3, random_state = 0)
```

[220]:
```
# 06.02.33.05
# preparing data for modeling
# assign regression variable
# dt15

dt15_lr01 = LinearRegression()
```

[221]:
```
# 06.02.33.06
# fit data for modeling
# fit variables to model
# dt15

dt15_lr01.fit(dt15_x01_trn, dt15_y01_trn)
```

[221]:
```
LinearRegression()
```

[222]:
```
# 06.02.33.07
# predict data for modeling
# fit variables to model
# dt15

dt15_y01_pdct = dt15_lr01.predict(dt15_x01_tst)
```

[223]:
```
# 06.02.33.08
# preparing data for modeling
# assign variable for rmse and r2
# dt15

dt15_rmse01 = np.sqrt(mean_squared_error(dt15_y01_tst, dt15_y01_pdct))
dt15_r201 = r2_score(dt15_y01_tst, dt15_y01_pdct)
```

[224]:
```
# 06.02.33.09
# run model
```

```
# return rmse and r2 dt15
# rmse: 2.4219084643113994e-14
# r2: 1.0
# dt15

print(f'rmse: {dt15_rmse01}')
print(f'r2: {dt15_r201}')
```

```
rmse: 2.4249800886174438e-14
r2: 1.0
```

[225]:
```
# 06.02.33.10
# assign variable for pca
# dt15

pca = PCA(.9)
```

[226]:
```
# 06.02.33.11
# calculate pca
# dt15

pca.fit(dt15_x01_trn)
dt15_x01_pca_trn = pca.transform(dt15_x01_trn)
dt15_x01_pca_tst = pca.transform(dt15_x01_tst)
```

[227]:
```
# 06.02.33.12
# return pca calculation matrix
# dt15

print(f'features in pca matrix: {dt15_x01_pca_trn.shape[1]}')
```

```
features in pca matrix: 1
```

[228]:
```
# 06.02.34.01
# preparing data for modeling
# add index column
# index column to select specific rows
# dt15

'''dt15_spatial_____________00_rn_drp.insert(0, 'index', range(0, 0 +␣
↪len(dt15_spatial_____________00_rn_drp)))'''
```

[228]:
```
"dt15_spatial_____________00_rn_drp.insert(0, 'index', range(0, 0 +
len(dt15_spatial_____________00_rn_drp)))"
```

[229]:
```
# 06.02.34.02
# preparing data for modeling
# create dummy variables
```

```
# due to returning boolean values, converting dummmies to integers
# dt15

'''dt15_spatial_____________________00_rn_drp_dv = pd.
 ↪get_dummies(dt15_spatial_____________________00_rn_drp, drop_first = True, dtype␣
 ↪= int)'''
```

[229]:
```
'dt15_spatial_____________________00_rn_drp_dv =
pd.get_dummies(dt15_spatial_____________________00_rn_drp, drop_first = True, dtype
= int)'
```

[230]:
```
# 06.02.34.03
# preparing data for modeling
# split data
# select columns
# dt15

dt15_x02 = dt15_spatial_____________________00_rn_drp_dv.drop(['humidity'], axis =␣
 ↪1)
dt15_y02 = dt15_spatial_____________________00_rn_drp_dv['humidity']
```

[231]:
```
# 06.02.34.04
# preparing data for modeling
# split into train and test
# dt15

dt15_x02_trn, dt15_x02_tst, dt15_y02_trn, dt15_y02_tst =␣
 ↪train_test_split(dt15_x02, dt15_y02, test_size = 0.3, random_state = 0)
```

[232]:
```
# 06.02.34.05
# preparing data for modeling
# assign regression variable
# dt15

dt15_lr02 = LinearRegression()
```

[233]:
```
# 06.02.34.06
# fit data for modeling
# fit variables to model
# dt15

dt15_lr02.fit(dt15_x02_trn, dt15_y02_trn)
```

[233]:
```
LinearRegression()
```

[234]:
```
# 06.02.34.07
# predict data for modeling
```

```
# fit variables to model
# dt15

dt15_y02_pdct = dt15_lr02.predict(dt15_x02_tst)
```

[235]:
```
# 06.02.34.08
# preparing data for modeling
# assign variable for rmse and r2
# dt15

dt15_rmse02 = np.sqrt(mean_squared_error(dt15_y02_tst, dt15_y02_pdct))
dt15_r202 = r2_score(dt15_y02_tst, dt15_y02_pdct)
```

[236]:
```
# 06.02.34.09
# run model
# return rmse and r2 dt15
# rmse: 14.341553065309144
# r2: 0.5739507710177247
# dt15

print(f'rmse: {dt15_rmse02}')
print(f'r2: {dt15_r202}')
```

```
rmse: 14.341553065309142
r2: 0.573950771017725
```

[237]:
```
# 06.02.34.10
# assign variable for pca
# dt15

pca = PCA(.9)
```

[238]:
```
# 06.02.34.11
# calculate pca
# dt15

pca.fit(dt15_x02_trn)
dt15_x02_pca_trn = pca.transform(dt15_x02_trn)
dt15_x02_pca_tst = pca.transform(dt15_x02_tst)
```

[239]:
```
# 06.02.34.12
# return pca calculation matrix
# dt15

print(f'features in pca matrix: {dt15_x02_pca_trn.shape[1]}')
```

```
features in pca matrix: 1
```

[240]:
```
# 06.02.35.01
# read csv
# assign variable
# dt16

dt16_emissions_____________00 = pd.read_csv('car_fuel_emissions.csv')
```

[241]:
```
# 06.02.35.02
# return first and last ten rows
# dt016

print(dt16_emissions_____________00.head(10))
print(dt16_emissions_____________00.tail(10))
```

```
                    file  year  … first_year_6_months date_of_change
0  DatapartC_july2000.csv  2000  …                 NaN            NaN
1  DatapartC_july2000.csv  2000  …                 NaN            NaN
2  DatapartC_july2000.csv  2000  …                 NaN            NaN
3  DatapartC_july2000.csv  2000  …                 NaN            NaN
4  DatapartC_july2000.csv  2000  …                 NaN            NaN
5  DatapartC_july2000.csv  2000  …                 NaN            NaN
6  DatapartC_july2000.csv  2000  …                 NaN            NaN
7  DatapartC_july2000.csv  2000  …                 NaN            NaN
8  DatapartC_july2000.csv  2000  …                 NaN            NaN
9  DatapartC_july2000.csv  2000  …                 NaN            NaN

[10 rows x 31 columns]
                                       file  …  date_of_change
45501  download-data-for-Aug-2013-Euro-6.csv  …             NaN
45502  download-data-for-Aug-2013-Euro-6.csv  …             NaN
45503  download-data-for-Aug-2013-Euro-6.csv  …             NaN
45504  download-data-for-Aug-2013-Euro-6.csv  …             NaN
45505  download-data-for-Aug-2013-Euro-6.csv  …             NaN
45506  download-data-for-Aug-2013-Euro-6.csv  …             NaN
45507  download-data-for-Aug-2013-Euro-6.csv  …             NaN
45508  download-data-for-Aug-2013-Euro-6.csv  …             NaN
45509  download-data-for-Aug-2013-Euro-6.csv  …             NaN
45510  download-data-for-Aug-2013-Euro-6.csv  …             NaN

[10 rows x 31 columns]
```

[242]:
```
# 06.02.35.03
# return dimensions
# dt16

print(dt16_emissions_____________00.shape)
```

```
(45511, 31)
```

[243]:
```
# 06.02.35.04
# confirm column names
# dt16

print(dt16_emissions_____________00.columns)
```

```
Index(['file', 'year', 'manufacturer', 'model', 'description', 'euro_standard',
       'tax_band', 'transmission', 'transmission_type', 'engine_capacity',
       'fuel_type', 'urban_metric', 'extra_urban_metric', 'combined_metric',
       'urban_imperial', 'extra_urban_imperial', 'combined_imperial',
       'noise_level', 'co2', 'thc_emissions', 'co_emissions', 'nox_emissions',
       'thc_nox_emissions', 'particulates_emissions', 'fuel_cost_12000_miles',
       'fuel_cost_6000_miles', 'standard_12_months', 'standard_6_months',
       'first_year_12_months', 'first_year_6_months', 'date_of_change'],
      dtype='object')
```

[244]:
```
# 06.02.35.05
# column rename to remove spaces and quotes
# dt16

dt16_emissions_____________00_rn = dt16_emissions_____________00.
 ↪rename(columns = {
    'file': 'file',
    'year': 'year',
    'manufacturer': 'manufacturer',
    'model': 'model',
    'description': 'description',
    'euro_standard': 'euro_standard',
    'tax_band': 'tax_band',
    'transmission': 'transmission',
    'transmission_type': 'transmission_type',
    'engine_capacity': 'engine_capacity',
    'fuel_type': 'fuel_type',
    'urban_metric': 'urban_metric',
    'extra_urban_metric': 'extra_urban_metric',
    'combined_metric': 'combined_metric',
    'urban_imperial': 'urban_imperial',
    'extra_urban_imperial': 'extra_urban_imperial',
    'combined_imperial': 'combined_imperial',
    'noise_level': 'noise_level',
    'co2': 'co2',
    'thc_emissions': 'thc_emissions',
    'co_emissions': 'co_emissions',
    'nox_emissions': 'nox_emissions',
    'thc_nox_emissions': 'thc_nox_emissions',
    'particulates_emissions': 'particulates_emissions',
    'fuel_cost_12000_miles': 'fuel_cost_12000_miles',
```

```
    'fuel_cost_6000_miles': 'fuel_cost_6000_miles',
    'standard_12_months': 'standard_12_months',
    'standard_6_months': 'standard_6_months',
    'first_year_12_months': '1st_year_12_months',
    'first_year_6_months': '1st_year_6_months',
    'date_of_change': 'date_change'
    })
```

[245]:
```
# 06.02.35.06
# confirm column names
# dt16

print(dt16_emissions_____________00_rn.columns)
```

```
Index(['file', 'year', 'manufacturer', 'model', 'description', 'euro_standard',
       'tax_band', 'transmission', 'transmission_type', 'engine_capacity',
       'fuel_type', 'urban_metric', 'extra_urban_metric', 'combined_metric',
       'urban_imperial', 'extra_urban_imperial', 'combined_imperial',
       'noise_level', 'co2', 'thc_emissions', 'co_emissions', 'nox_emissions',
       'thc_nox_emissions', 'particulates_emissions', 'fuel_cost_12000_miles',
       'fuel_cost_6000_miles', 'standard_12_months', 'standard_6_months',
       '1st_year_12_months', '1st_year_6_months', 'date_change'],
      dtype='object')
```

[246]:
```
# 06.02.35.07
# confirm types
# dt16

print(dt16_emissions_____________00_rn['file'].dtypes)
print(dt16_emissions_____________00_rn['year'].dtypes)
print(dt16_emissions_____________00_rn['manufacturer'].dtypes)
print(dt16_emissions_____________00_rn['model'].dtypes)
print(dt16_emissions_____________00_rn['description'].dtypes)
print(dt16_emissions_____________00_rn['euro_standard'].dtypes)
print(dt16_emissions_____________00_rn['tax_band'].dtypes)
print(dt16_emissions_____________00_rn['transmission'].dtypes)
print(dt16_emissions_____________00_rn['transmission_type'].dtypes)
print(dt16_emissions_____________00_rn['engine_capacity'].dtypes)
print(dt16_emissions_____________00_rn['fuel_type'].dtypes)
print(dt16_emissions_____________00_rn['urban_metric'].dtypes)
print(dt16_emissions_____________00_rn['extra_urban_metric'].dtypes)
print(dt16_emissions_____________00_rn['combined_metric'].dtypes)
print(dt16_emissions_____________00_rn['urban_imperial'].dtypes)
print(dt16_emissions_____________00_rn['extra_urban_imperial'].dtypes)
print(dt16_emissions_____________00_rn['combined_imperial'].dtypes)
print(dt16_emissions_____________00_rn['noise_level'].dtypes)
print(dt16_emissions_____________00_rn['co2'].dtypes)
print(dt16_emissions_____________00_rn['thc_emissions'].dtypes)
```

```
print(dt16_emissions_____________00_rn['co_emissions'].dtypes)
print(dt16_emissions_____________00_rn['nox_emissions'].dtypes)
print(dt16_emissions_____________00_rn['thc_nox_emissions'].dtypes)
print(dt16_emissions_____________00_rn['particulates_emissions'].dtypes)
print(dt16_emissions_____________00_rn['fuel_cost_12000_miles'].dtypes)
print(dt16_emissions_____________00_rn['fuel_cost_6000_miles'].dtypes)
print(dt16_emissions_____________00_rn['standard_12_months'].dtypes)
print(dt16_emissions_____________00_rn['standard_6_months'].dtypes)
print(dt16_emissions_____________00_rn['1st_year_12_months'].dtypes)
print(dt16_emissions_____________00_rn['1st_year_6_months'].dtypes)
print(dt16_emissions_____________00_rn['date_change'].dtypes)
```

```
object
int64
object
object
object
int64
object
object
object
float64
object
float64
float64
float64
float64
float64
float64
float64
int64
float64
float64
float64
float64
float64
float64
float64
float64
float64
float64
float64
object
```

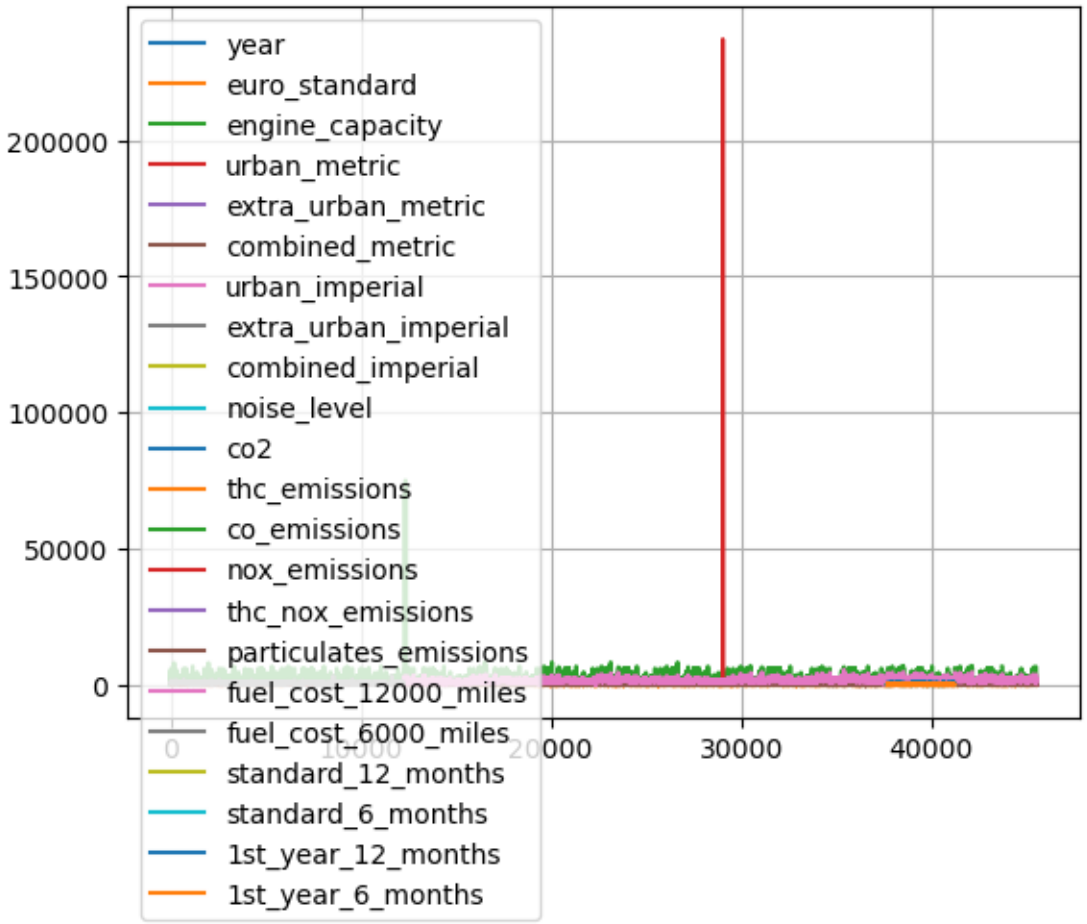
[247]:
```
# 06.02.35.08
# change types
# dt16
```

```
'''dt16_emissions_____________00_rn[''].astype('float')'''
```

[247]:
```
"dt16_emissions_____________00_rn[''].astype('float')"
```

[248]:
```
# 06.02.35.09
# return basic plot
# rendered basic plot to see visually and to determine if data is numeric
# dt16

plt.figure(figsize=(18,8))
dt16_emissions_____________00_rn.plot()
plt.box(True)
plt.grid(True)
plt.title('', fontsize = 16, color = '#0047ab')
plt.xlabel('', fontsize = 14, color = '#0047ab')
plt.ylabel('', fontsize = 14, color = '#0047ab')
plt.show()
```

```
<Figure size 1800x800 with 0 Axes>
```

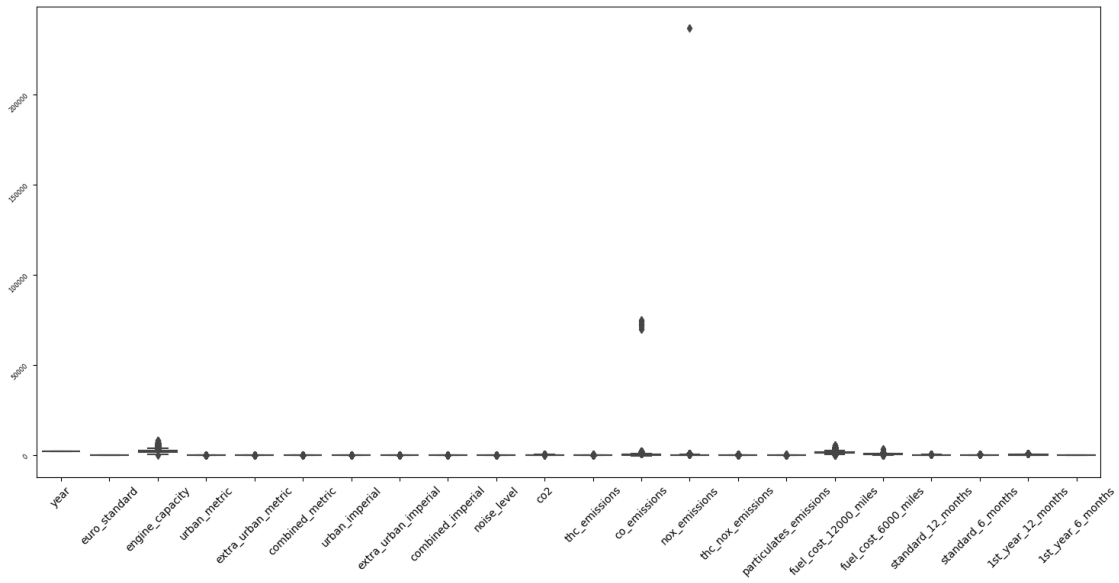


[249]:
```
# 06.02.35.10
# render bar chart
# dt16

'''plt.figure(figsize=(18,8))
dt16_emissions_____________00_rn.plot(kind='bar', stacked=True)
plt.title('Air Quality')
plt.xticks(rotation=0, ha='center')
plt.show()'''
```

[249]:
```
"plt.figure(figsize=(18,8))\ndt16_emissions_____________00_rn.plot(kind='bar',
stacked=True)\nplt.title('Air Quality')\nplt.xticks(rotation=0,
ha='center')\nplt.show()"
```

[250]:
```
# 06.02.35.11
# return columns
# dt16

plt.figure(figsize=(18,8))
sns.boxplot(dt16_emissions_____________00_rn)
plt.xticks(fontsize = 10)
plt.xticks(rotation = 45)
plt.yticks(fontsize = 6)
plt.yticks(rotation = 45)
```

[250]:
```
(array([-50000.,      0.,  50000., 100000., 150000., 200000., 250000.]),
 [Text(0, -50000.0, '-50000'),
  Text(0, 0.0, '0'),
  Text(0, 50000.0, '50000'),
  Text(0, 100000.0, '100000'),
  Text(0, 150000.0, '150000'),
  Text(0, 200000.0, '200000'),
  Text(0, 250000.0, '250000')])
```



[251]:
```
# 06.02.35.12
# replace NaNs with mean
# dt16

'''val_mean = dt16_emissions_____________00_rn['co_emissions'].mean()
dt16_emissions_____________00_rn['co_emissions'].fillna(value=val_mean,␣
 ↪inplace=True)
print('Updated Dataframe:')
print(dt16_emissions_____________00_rn)'''
```

[251]:
```
"val_mean = dt16_emissions_____________00_rn['co_emissions'].mean()
\ndt16_emissions_____________00_rn['co_emissions'].fillna(value=val_mean,
inplace=True) \nprint('Updated
Dataframe:')\nprint(dt16_emissions_____________00_rn)"
```

[252]:
```
# 06.02.35.13
# replace NaNs with mean
# dt16

'''val_mean = dt16_emissions_____________00_rn.mean()
dt16_emissions_____________00_rn.fillna(value=val_mean, inplace=True)
print('Updated Dataframe:')
print(dt16_emissions_____________00_rn) '''
```

[252]:
```
"val_mean = dt16_emissions_____________00_rn.mean()
\ndt16_emissions_____________00_rn.fillna(value=val_mean, inplace=True)
\nprint('Updated Dataframe:')\nprint(dt16_emissions_____________00_rn) "
```

[253]:
```
# 06.02.36.01
# function async
# dt16

def func_async(i, *args):
    return 2 * i

print(Parallel(n_jobs=2)(delayed(func_async)(21) for _ in range(1))[0])
```

```
42
```

[254]:
```
# 06.02.36.02
# function async
# pass an extra argument with a large list
# dt16

'''def func_async(i, *args):
    return 2 * i

dt16_emissions_____________00_rn_lst =␣
 ↪dt16_emissions_____________00_rn(range(1000000))

t_start = time.time()
Parallel(n_jobs=2)(delayed(func_async)(21,␣
 ↪dt16_emissions_____________00_rn_lst) for _ in range(1))
print("With loky backend and cloudpickle serialization: {:.3f}s"
      .format(time.time() - t_start))'''
```

[254]:
```
'def func_async(i, *args):\n    return 2 *
i\n\ndt16_emissions_____________00_rn_lst =
dt16_emissions_____________00_rn(range(1000000))\n\nt_start =
time.time()\nParallel(n_jobs=2)(delayed(func_async)(21,
dt16_emissions_____________00_rn_lst) for _ in range(1))\nprint("With loky
backend and cloudpickle serialization: {:.3f}s"\n      .format(time.time() -
t_start))'
```

[255]:
```
# 06.02.36.03
# function mp
# dt16

if mp.get_start_method() != "spawn":
    def func_async(i, *args):
        return 2 * i

    with parallel_config('multiprocessing'):
        t_start = time.time()
        Parallel(n_jobs=2)(
```

```
            delayed(func_async)(21, dt16_emissions_____________00_rn_lst) for␣
 ↪_ in range(1))
        print("With multiprocessing backend and pickle serialization: {:.3f}s"
              .format(time.time() - t_start))
```

[256]:
```
# 06.02.36.04
# set the `loky_pickler` to use pickle serialization from stdlib
# do not pass desired function ``func_async`` as it is not picklable
# replaced by ``id`` for demonstration purposes
# dt16

set_loky_pickler('pickle')
t_start = time.time()
Parallel(n_jobs=2)(delayed(id)(dt16_emissions_____________00_rn) for _ in␣
 ↪range(1))
print("With pickle serialization: {:.3f}s".format(time.time() - t_start))
```

```
With pickle serialization: 2.762s
```

[257]:
```
# 06.02.36.05
# function async
# dt16

def func_async(i, *args):
    return 2 * i

try:
    Parallel(n_jobs=2)(delayed(func_async)(21,␣
 ↪dt16_emissions_____________00_rn) for _ in range(1))
except Exception:
    traceback.print_exc(file=sys.stdout)
```

```
joblib.externals.loky.process_executor._RemoteTraceback:
"""
Traceback (most recent call last):
  File "/Users/gimjun-won/anaconda3/lib/python3.11/site-
packages/joblib/externals/loky/process_executor.py", line 391, in
_process_worker
    call_item = call_queue.get(block=True, timeout=timeout)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/gimjun-won/anaconda3/lib/python3.11/multiprocessing/queues.py",
line 122, in get
    return _ForkingPickler.loads(res)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^
AttributeError: Can't get attribute 'func_async' on <module
'joblib.externals.loky.backend.popen_loky_posix' from '/Users/gimjun-
won/anaconda3/lib/python3.11/site-
packages/joblib/externals/loky/backend/popen_loky_posix.py'>
```

```
"""

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/folders/vb/3rppqxbs6y589jc4_y00n23h0000gn/T/ipykernel_54362/3245748
660.py", line 9, in <module>
    Parallel(n_jobs=2)(delayed(func_async)(21,
dt16_emissions_____________00_rn) for _ in range(1))
  File "/Users/gimjun-won/anaconda3/lib/python3.11/site-
packages/joblib/parallel.py", line 1098, in __call__
    self.retrieve()
  File "/Users/gimjun-won/anaconda3/lib/python3.11/site-
packages/joblib/parallel.py", line 975, in retrieve
    self._output.extend(job.get(timeout=self.timeout))
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/gimjun-won/anaconda3/lib/python3.11/site-
packages/joblib/_parallel_backends.py", line 567, in wrap_future_result
    return future.result(timeout=timeout)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/gimjun-won/anaconda3/lib/python3.11/concurrent/futures/_base.py",
line 456, in result
    return self.__get_result()
           ^^^^^^^^^^^^^^^^^^^
  File "/Users/gimjun-won/anaconda3/lib/python3.11/concurrent/futures/_base.py",
line 401, in __get_result
    raise self._exception
joblib.externals.loky.process_executor.BrokenProcessPool: A task has failed to
un-serialize. Please ensure that the arguments of the function are all
picklable.
```

[258]:
```
# 06.02.36.06
# function async wrapped
# dt16

@delayed
@wrap_non_picklable_objects
def func_async_wrapped(i, *args):
    return 2 * i

t_start = time.time()
Parallel(n_jobs=2)(func_async_wrapped(21, dt16_emissions_____________00_rn)␣
 ↪for _ in range(1))
print("With pickle from stdlib and wrapper: {:.3f}s"
      .format(time.time() - t_start))
```

```
With pickle from stdlib and wrapper: 1.692s
```

[259]:
```
# 06.02.36.07
# Reset loky_pickler to avoid border effects
# dt16

set_loky_pickler()
```

[260]:
```
# 06.02.37.01
# preparing data for modeling
# add index column
# index column to select specific rows
# dt16

dt16_emissions_____________00_rn.insert(0, 'index', range(0, 0 +␣
 ↪len(dt16_emissions_____________00_rn)))
```

[261]:
```
# 06.02.37.02
# preparing data for modeling
# create dummy variables
# due to returning boolean values, converting dummmies to integers
# dt16

dt16_emissions_____________00_rn_dv = pd.
 ↪get_dummies(dt16_emissions_____________00_rn, drop_first = True, dtype =␣
 ↪int)
```

[262]:
```
# 06.02.37.03
# preparing data for modeling
# split data
# select columns
# dt16

dt16_x01 = dt16_emissions_____________00_rn_dv.drop(['co_emissions'], axis =␣
 ↪1)
dt16_y01 = dt16_emissions_____________00_rn_dv['co_emissions']
```

[263]:
```
# 06.02.37.04
# preparing data for modeling
# split into train and test
# dt16

'''dt16_x01_trn, dt16_x01_tst, dt16_y01_trn, dt16_y01_tst =␣
 ↪train_test_split(dt16_x01, dt16_y01, test_size = 0.3, random_state = 0)'''
```

[263]:
```
'dt16_x01_trn, dt16_x01_tst, dt16_y01_trn, dt16_y01_tst =
train_test_split(dt16_x01, dt16_y01, test_size = 0.3, random_state = 0)'
```

[264]:
```
# 06.02.37.05
# preparing data for modeling
# assign regression variable
# dt16

'''dt16_lr01 = LinearRegression()'''
```

[264]:
```
'dt16_lr01 = LinearRegression()'
```

[265]:
```
# 06.02.37.06
# fit data for modeling
# fit variables to model
# dt16

'''dt16_lr01.fit(dt16_x01_trn, dt16_y01_trn)'''
```

[265]:
```
'dt16_lr01.fit(dt16_x01_trn, dt16_y01_trn)'
```

[266]:
```
# 06.02.37.07
# predict data for modeling
# fit variables to model
# dt16

'''dt16_y01_pdct = dt16_lr01.predict(dt16_x01_tst)'''
```

[266]:
```
'dt16_y01_pdct = dt16_lr01.predict(dt16_x01_tst)'
```

[267]:
```
# 06.02.37.08
# preparing data for modeling
# assign variable for rmse and r2
# dt16

'''dt16_rmse01 = np.sqrt(mean_squared_error(dt16_y01_tst, dt16_y01_pdct))
dt16_r201 = r2_score(dt16_y01_tst, dt16_y01_pdct)'''
```

[267]:
```
'dt16_rmse01 = np.sqrt(mean_squared_error(dt16_y01_tst,
dt16_y01_pdct))\ndt16_r201 = r2_score(dt16_y01_tst, dt16_y01_pdct)'
```

[268]:
```
# 06.02.37.09
# run model
# return rmse and r2 dt16
# rmse: 2.4219084643113994e-14
# r2: 1.0
# dt16

'''print(f'rmse: {dt16_rmse01}')
print(f'r2: {dt16_r201}')'''
```

[268]:
```
"print(f'rmse: {dt16_rmse01}')\nprint(f'r2: {dt16_r201}')"
```

[269]:
```
# 06.02.37.10
# assign variable for pca
# dt16

'''pca = PCA(.9)'''
```

[269]:
```
'pca = PCA(.9)'
```

[270]:
```
# 06.02.37.11
# calculate pca
# dt16

'''pca.fit(dt16_x01_trn)
dt16_x01_pca_trn = pca.transform(dt16_x01_trn)
dt16_x01_pca_tst = pca.transform(dt16_x01_tst)'''
```

[270]:
```
'pca.fit(dt16_x01_trn)\ndt16_x01_pca_trn =
pca.transform(dt16_x01_trn)\ndt16_x01_pca_tst = pca.transform(dt16_x01_tst)'
```

[271]:
```
# 06.02.37.12
# return pca calculation matrix
# dt16

'''print(f'features in pca matrix: {dt16_x01_pca_trn.shape[1]}')'''
```

[271]:
```
"print(f'features in pca matrix: {dt16_x01_pca_trn.shape[1]}')"
```

[272]:
```
# 06.02.38.01
# assign variables for classification
# Accuracy: 0.944
# dt16

dt16_x01, dt16_y01 = make_classification(n_samples=10000, n_features=100,␣
 ↪n_informative=50, n_redundant=50, random_state=1)
dt16_x01_trn, dt16_x01_tst, dt16_y01_trn, dt16_y01_tst =␣
 ↪train_test_split(dt16_x01, dt16_y01, test_size=0.2, random_state=42)
dt16_model = HistGradientBoostingClassifier(max_bins=255, max_iter=100)
dt16_model.fit(dt16_x01_trn, dt16_y01_trn)
dt16_y01_pdct = dt16_model.predict(dt16_x01_tst)
dt16_accuracy = accuracy_score(dt16_y01_tst, dt16_y01_pdct)
print(f'Accuracy: {dt16_accuracy:.3f}')
```

```
Accuracy: 0.944
```