DataPreprocessing 数据预处理

文件结构：

1. **数据清洗** /DataCleaning
    1. 空值处理 MissingDataHandle.py
        1. 删除 delete_handle(data_class, handel_index)
        2. 中位数插补 median_interpolation_handle(data_class, handel_index)
        3. 众数插补 mode_interpolation_handle(data_class, handel_index)
        4. 均值插补 mean_interpolation_handle(data_class, handel_index)
        5. 固定值插补 fixed_value_padding_handle(data_class, handel_index, padding_value)
        6. 间值法插补 mid_interpolation_handle(data_class, handel_index)
        7. 线性回归法
        8. 拉格朗日法
    2. 异常值 OutlierHandle.py
        1. Z-Score法 z_score_detection(data_class, handel_index, z_thr=3.0)
2. **数据集成** /Discretization
    1. 合并
    2. 去重
3. **数据变换** /DataTransformation
    1. 函数变换
    2. 归一化 NormalizeHandle.py
        1. 离差标准化 min_max_normalize(data_class, handel_index)
        2. 反离差标准化 anti_min_max_normalize(data_class, handel_index)
    3. 标准化 StandardizationHandle.py
        1. 标准化 standardization(data_class, handel_index)
        2. 反标准化 anti_standardization(data_class, handel_index)
4. **数据规约** /DataReduction
5. **数据离散** /DataIntegration
6. **数据集结构** DataClass.py
    1. 数据读取 read(self, path, has_head, split_tag='\t')
    2. 数据格式转换 parse(self)
7. **日志记录** LogHelper.py

一、数据表结构DataClass

包括属性:

1. 二维的数据表 `data = [[]]`
2. 表头 `head`
3. 每一列的数据类型 `type_list`
4. 归一化时的最大值列表（用于反归一化） `normalize_max`
5. 归一化时的最小值列表（用于反归一化） `normalize_min`
6. 标准化时的均值（用于反标准化） `standard_mean`
7. 标准化时的标准差（用于反标准化） `standard_std`

包括方法：

- 数据读取 `read(self, path, has_head, split_tag='\t'):`
    - path:文件路径
    - has_head：是否有表头
    - split_tag：切分字符
- 数据格式转换 `parse(self)`

```
data = DataClass([str] + [float] * 12)
data.read(r".\sample\fz_micro.txt", True)
data.parse()
```

.\sample\fz_micro.txt（部分）

RECEIVETIME	CO	NO2	SO2	O3	PM25	PM10	TEMP	HUM	PM05N	PM1N	PM25N	PM10N
2017/1/9 18:00		634.38	619.43	733.52	57.33	57.76	15.19	65.14	4026.38	1944.57	401.29	24.81
2017/1/9 19:00	431.47	962.93	570.17	824.27	51.8	52.17	14	67.8	3646.73	1758.57	357.47	22.6
2017/1/9 20:00	423	756.33	556.43	854.57	48.57	48.73	14	68.3	3513	1687.4	339.77	20.7
2017/1/9 21:00	419.93	1008.57	499.47	908.13	46	46.47	13.8	68.33	3345.13	1600.43	326.17	20.87
2017/1/9 22:00		1019.47	476.07	927.67	46.27	46.77	13.03	68.83	3401.73	1633.37	328.7	18.83
2017/1/9 23:00		904.8	475.37	947.03	53.47	53.8	13	68.63	3838.1	1856.47	379.37	22.07
2017/1/10 0:00	412.9	1052.7	467.23	955.4	60.5	60.87	**A5**	68.3	4242.37	2075.77	428.53	25.93
2017/1/10 1:00	412.93	876.2	503.9	930.7	66.8	67.17	13	68.07	4635.37			

二、数据清洗 /DataCleaning

1. 空值处理 MissingDataHandle.py

1.1 空值删除 delete_handle(data_class, handel_index)

1. `data_class` 类型为DataClass的数据
2. `handel_index` 要处理的列的下标

```
import DataClass as dc

data = dc.DataClass([str] + [float] * 12)
data.read(r".\sample\fz_micro.txt", False)
delete_handle(data,[i for i in range(1, 13)])
data.parse()
```

处理后的 .data (空值删除并不会检查数据类型是否合法，如**A5**并不会被删除)

RECEIVETIME	CO	NO2	SO2	O3	PM25	PM10	TEMP	HUM	PM05N	PM1N	PM25N	PM10N
2017/1/9 19:00	431.47	962.93	570.17	824.27	51.8	52.17	14	67.8	3646.73	1758.57	357.47	22.6
2017/1/9 20:00	423	756.33	556.43	854.57	48.57	48.73	14	68.3	3513	1687.4	339.77	20.7
2017/1/9 21:00	419.93	1008.57	499.47	908.13	46	46.47	13.8	68.33	3345.13	1600.43	326.17	20.87
2017/1/10 0:00	412.9	1052.7	467.23	955.4	60.5	60.87	**A5**	68.3	4242.37	2075.77	428.53	25.93

1.2 均数填充 mean_interpolation_handle(data_class, handel_index)

要处理的属性必须是数值的，**不是数值元素按空值处理**

处理后的 .data

RECEIVETIME	CO	NO2	SO2	O3	PM25	PM10	TEMP	HUM	PM05N	PM1N	PM25N	PM10N
2017/1/9 18:00	**411.02**	634.38	619.43	733.52	57.33	57.76	15.19	65.14	4026.38	1944.57	401.29	24.81
2017/1/9 19:00	431.47	962.93	570.17	824.27	51.8	52.17	14	67.8	3646.73	1758.57	357.47	22.6
2017/1/9 20:00	423	756.33	556.43	854.57	48.57	48.73	14	68.3	3513	1687.4	339.77	20.7
2017/1/9 21:00	419.93	1008.57	499.47	908.13	46	46.47	13.8	68.33	3345.13	1600.43	326.17	20.87
2017/1/9 22:00	**411.02**	1019.47	476.07	927.67	46.27	46.77	13.03	68.83	3401.73	1633.37	328.7	18.83
2017/1/9 23:00	**411.02**	904.8	475.37	947.03	53.47	53.8	13	68.63	3838.1	1856.47	379.37	22.07
2017/1/10 0:00	412.9	1052.7	467.23	955.4	60.5	60.87	**13.28**	68.3	4242.37	2075.77	428.53	25.93
2017/1/10 1:00	412.93	876.2	503.9	930.7	66.8	67.17	13	68.07	4635.37	**2279.67**	**469.8**	**28.93**

1.3 插值法填充 mid_interpolation_handle(data_class, handel_index)

要处理的属性必须是数值的，**不是数值元素按空值处理**.

1. 若空值处于首位，则插值取空值的下一个最近的非空的元素.
2. 若空值位于末尾，则插值取空值的上一个最近的非空的元素.
3. 若一个或多个连续的空值位于前后两个非空元素之间，则差值取前后非空元素的等差间值.

处理后的 .data

RECEIVETIME	CO	NO2	SO2	O3	PM25	PM10	TEMP	HUM	PM05N	PM1N	PM25N	PM10N
2017/1/9 18:00	**431.47**	634.38	619.43	733.52	57.33	57.76	15.19	65.14	4026.38	1944.57	401.29	24.81
2017/1/9 19:00	431.47	962.93	570.17	824.27	51.8	52.17	14	67.8	3646.73	1758.57	357.47	22.6
2017/1/9 20:00	423	756.33	556.43	854.57	48.57	48.73	14	68.3	3513	1687.4	339.77	20.7
2017/1/9 21:00	419.93	1008.57	499.47	908.13	46	46.47	13.8	68.33	3345.13	1600.43	326.17	20.87
2017/1/9 22:00	**417.58**	1019.47	476.07	927.67	46.27	46.77	13.03	68.83	3401.73	1633.37	328.7	18.83
2017/1/9 23:00	**415.24**	904.8	475.37	947.03	53.47	53.8	13	68.63	3838.1	1856.47	379.37	22.07
2017/1/10 0:00	412.9	1052.7	467.23	955.4	60.5	60.87	**13**	68.3	4242.37	2075.77	428.53	25.93
2017/1/10 1:00	412.93	876.2	503.9	930.7	66.8	67.17	13	68.07	4635.37	**2360.77**	**489.71**	**29.58**

1.4+ 中数填充 众数填充 固定值填充 等

2. 离异值（异常值）处理 OutlierHandle.py

2.1 Z-Score异常值检测 z_score_detection(data_class, handel_index, z_thr=3.0)

1. `data_class` 类型为DataClass的数据.
2. `handel_index` 要处理的列的下标.
3. `z_thr` 识别阈值. 一般取 2.5, 3.0, 3.5
4. `:return` 每一列离异值的下标.

条件：-1. 数据无空值. -2. 数据经过 parse() 方法格式转换.

```
import DataClass as dc
import DataCleaning.MissingDataHandle as mdh
import DataCleaning.OutlierHandle as oh

data = dc.DataClass([str] + [float] * 12)
data.read(r".\sample\fz_micro.txt", False)
data.parse()
mdh.mid_interpolation_handle(data, [i for i in range(1, 13)])
oh.outlier_none_handle(data, [i for i in range(1, 13)], "z_score", 3.0)
print(data.data)
```

三、数据变换 /DataTransformation

1. 归一化 NormalizeHandle.py

1.1 离差标准化 min_max_normalize(data_class, handel_index)

```
import DataClass as dc
import DataCleaning.MissingDataHandle as mdh
import DataTransformation.NormalizeHandle as nh

    data = dc.DataClass([str] + [float] * 12)
    data.read(r".\sample\fz_micro.txt", False)
    data.parse()
    mdh.mid_interpolation_handle(data, [i for i in range(1, 13)])
    nh.min_max_normalize(data, [i for i in range(1, 13)])
    for line in data.data:
        print('|'.join(['{:.2}'.format(line[i]) if i > 0 else line[i] for i in range(len(line))]))
```

条件：-1. 数据无空值. -2. 数据经过 parse() 方法格式转换处理. 处理后的 .data

RECEIVETIME	CO	NO2	SO2	O3	PM25	PM10	TEMP	HUM	PM05N	PM1N	PM25N	PM10N
2017/1/9 18:00	0.87	0.27	0.83	0.31	0.48	0.48	0.55	0.52	0.54	0.51	0.45	0.36
2017/1/9 19:00	0.87	0.51	0.76	0.44	0.43	0.43	0.43	0.57	0.49	0.46	0.4	0.33
2017/1/9 20:00	0.71	0.36	0.74	0.48	0.4	0.4	0.43	0.58	0.47	0.44	0.37	0.3
2017/1/9 21:00	0.66	0.55	0.66	0.56	0.38	0.38	0.41	0.58	0.44	0.41	0.36	0.3
2017/1/9 22:00	0.61	0.56	0.63	0.59	0.38	0.38	0.34	0.59	0.45	0.42	0.36	0.27
2017/1/9 23:00	0.57	0.47	0.63	0.61	0.45	0.44	0.33	0.59	0.52	0.48	0.42	0.32
2017/1/10 0:00	0.52	0.58	0.61	0.63	0.51	0.5	0.33	0.58	0.58	0.54	0.48	0.38
2017/1/10 1:00	0.53	0.45	0.67	0.59	0.56	0.56	0.33	0.58	0.63	0.62	0.55	0.44