```python
# from .init import search_blue
from flask import request
import pymysql
from dtw import *

# 创建连接
db = pymysql.connect(host='localhost', user='tester', password='test', port=3306)
# 创建游标
cursor = db.cursor()


# 相似度比对
# @search_blue.route("/similarity_search", methods=['POST'])
def similarity_search():
    input_json = request.get_json(force=True)
    print(input_json)
    rank = int(input_json['rank'])  # top10=>rank=10  top5=>rank=5
    ab_node = str(input_json['ab_node'])    # 异常数据表名
    start_time = str(input_json['start_time'])  # 开始时间
    end_time = str(input_json['end_time'])  # 截止时间
    # 相似度分析
    res = process_similarity(rank, ab_node, start_time, end_time)
    #todo: 处理结果
    return res


# 相似度分析函数
def process_similarity(rank: int, ab_node:str, start_time: str, end_time: str):
    # 返回值：根据rank返回一个list，元素为相似度从高到低的<node_name:simi_value>
    # simi_value：dtw算法算出的距离，距离越大表示相似度越低

    # 得到表名
    nodes_name = get_similarity_node()
    # 得到异常序列和对比序列列表
    ret = get_similarity_data(nodes_name, ab_node, start_time, end_time)
    nodes_name.remove(ab_node)
    query = ret.pop(ab_node)
    refers = ret

    simi_list = []
    while(len(refers)):
        curr_node = nodes_name.pop()
        refer = refers.pop(curr_node)
        # dtw算法对比两个序列的相似度
        alignment = dtw(x=query['y_data_list'],
                        y=refer['y_data_list'],
                        dist_method="euclidean",
                        step_pattern=asymmetric,
                        keep_internals=True)
        alignment.plot(type="twoway", offset=1, ylab=curr_node)
        # 将相似度分析结果加入list
        node_simi = dict()
        node_simi['node_name'] = curr_node
        node_simi['simi_value'] = alignment.__getattribute__('distance')
```

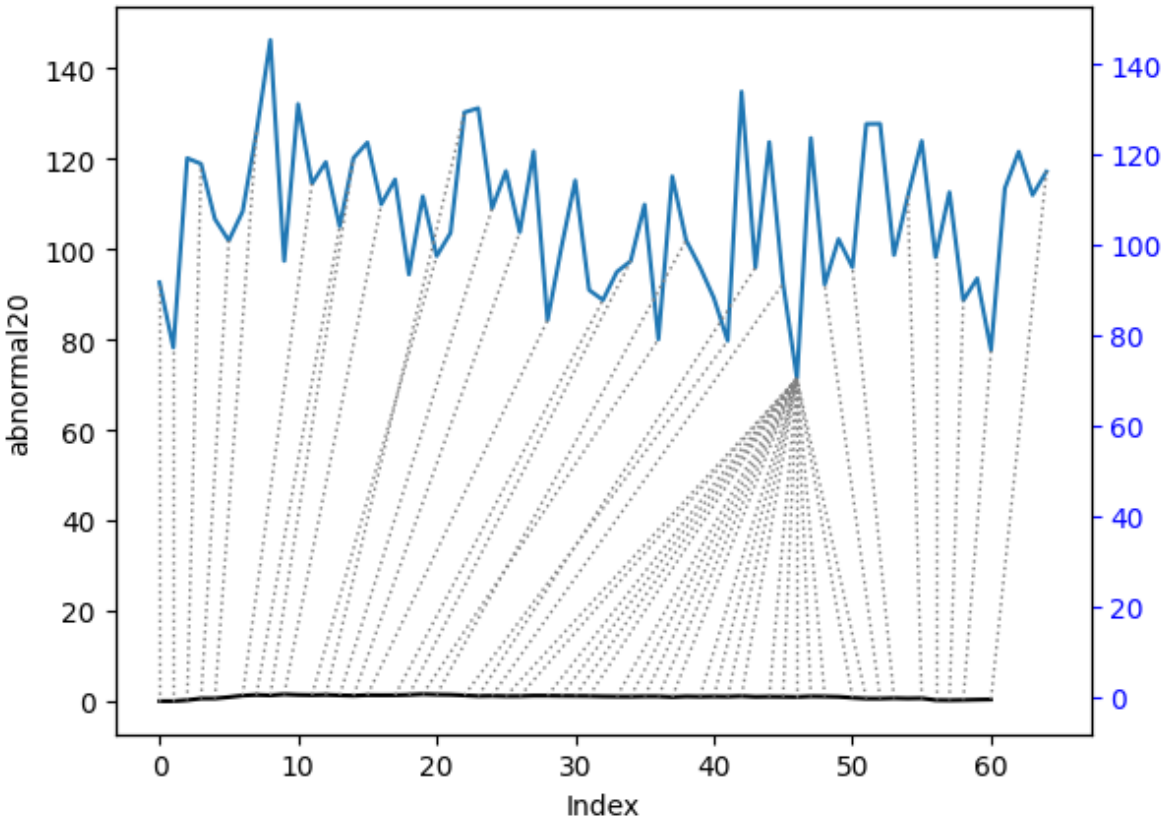
```
            simi_list.append(node_simi)
        # 按相似度大小排序，只返回rank个最相似序列
        simi_list.sort(key=lambda e:e['simi_value'])
        simi_list = simi_list[:rank]
        print("相似度排序：")
        for simi in simi_list:
            print(simi)
        return


# 获取相似度比对的数据表名
def get_similarity_node(filename="AIBMA-ES-CLUSTER-20230423"):
    database_name = filename.replace("-", "_")
    database_sql = "USE " + database_name
    cursor.execute(database_sql)
    db.commit()
    res = []
    # 查询表名
    sql = "select table_name from information_schema.tables where
table_schema='AIBMA_ES_CLUSTER_20230423' order by
cast(substr(table_name,9,length(table_name)) as signed ) asc;"
    cursor.execute(sql)
    ret = cursor.fetchall()
    for item in ret:
        res.append(item[0])
    return res


# 根据输入的表名获取相似度信息
def get_similarity_data(nodes_name:list, ab_node:str, start_time:str,
end_time:str):
    database_sql = "USE AIBMA_ES_CLUSTER_20230423"
    cursor.execute(database_sql)
    db.commit()
    res = dict()
    # 根据需要查询的节点名称去查询
    for node_name in nodes_name:
        res[node_name] = dict()
        datas = select_node_similarity_data(node_name, ab_node == node_name,
start_time, end_time)
        res[node_name]['y_data_list'] = datas
        res[node_name]['simi_value'] = 0
    return res


# 从表中截取序列
def select_node_similarity_data(table_name:str, isQuery:bool, start_time:str,
end_time:str):
    # 找到开始序列的首尾id
    sql = "select id from " + table_name + " where date=\'" + start_time +"\' or
date=\'" + end_time +"\'"
    cursor.execute(sql)
    datas = cursor.fetchall()
    start_id, end_id = datas[0][0], datas[1][0]

    sql = "select id,value from " + table_name + " where date!=''"
```

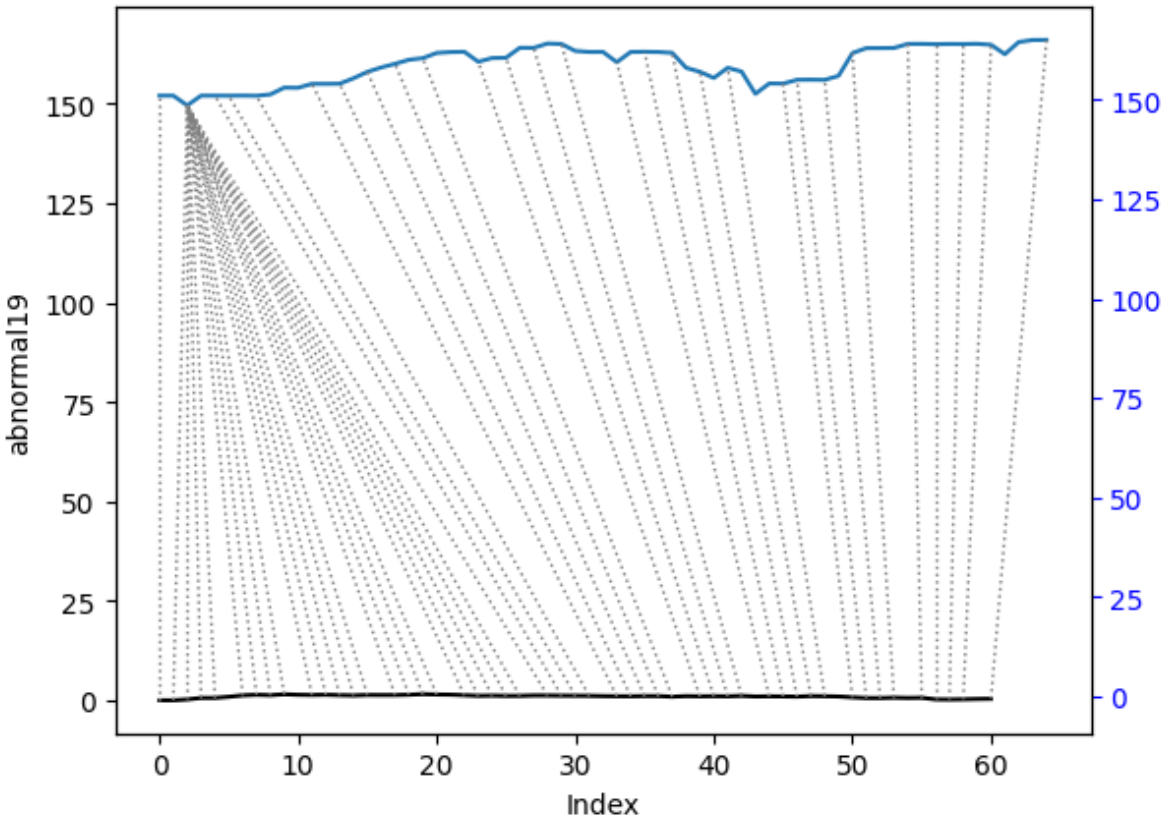
```
    cursor.execute(sql)
    datas = cursor.fetchall()
    y_data_list = []
    # 对于对比序列，考虑波动提前/延后的情况，将对比时间段边界放宽
    for data in datas:
        if(not isQuery and data[0] >= start_id-2 and data[0] < start_id):
            y_data_list.append(float(data[1]))
        elif(data[0] >= start_id and data[0] <= end_id):
            y_data_list.append(float(data[1]))
        elif(not isQuery and data[0] > end_id and data[0] <= end_id + 2):
            y_data_list.append(float(data[1]))
        elif(data[0] > end_id+2):
            break
    return y_data_list
```

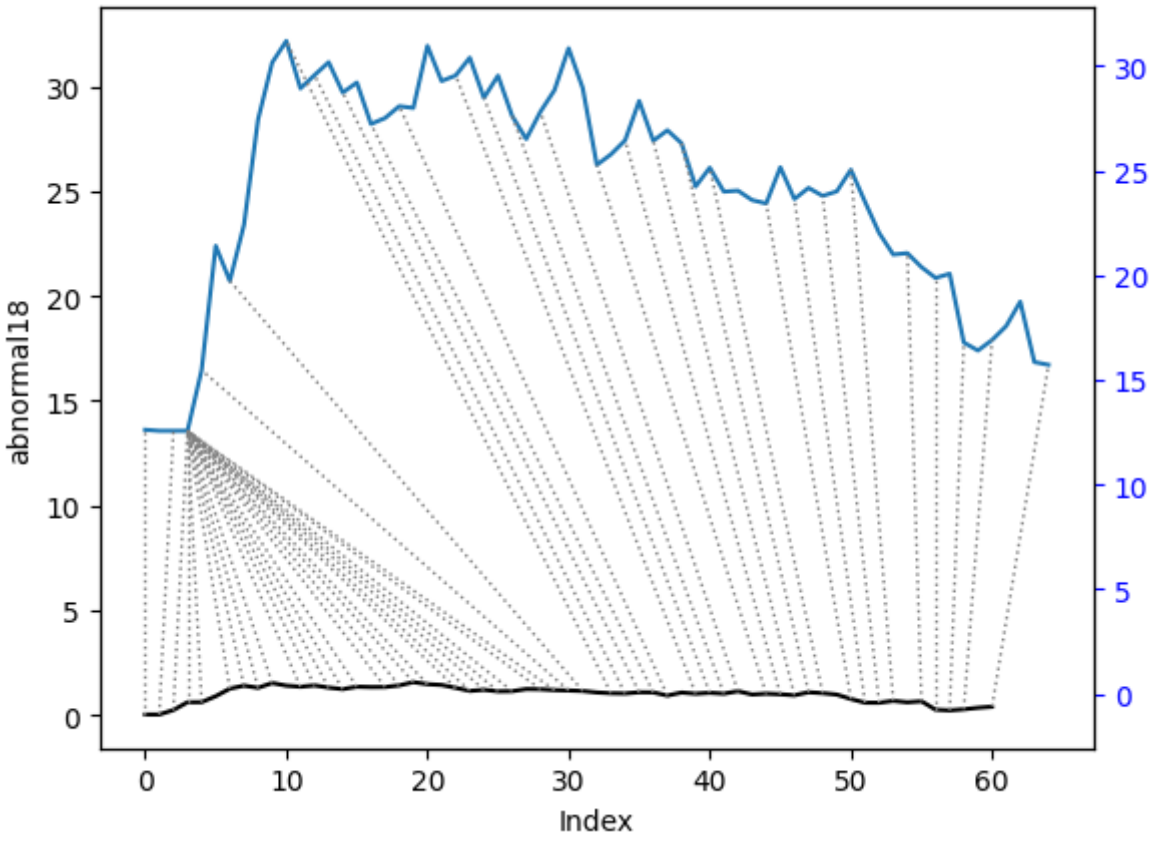
```
Importing the dtw module. When using in academic works please cite:
  T. Giorgino. Computing and Visualizing Dynamic Time Warping Alignments in R:
The dtw Package.
  J. Stat. Soft., doi:10.18637/jss.v031.i07.
```

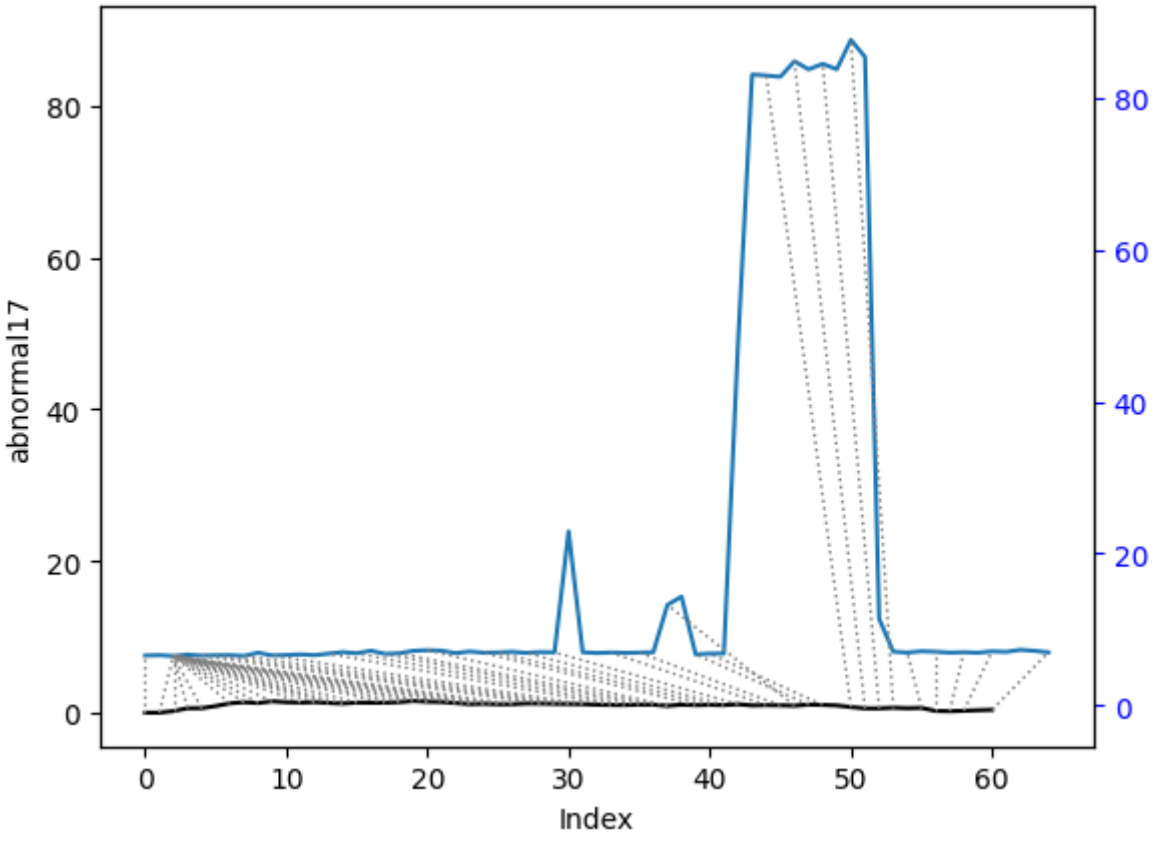
```
process_similarity(20, 'abnormal0', '2023/4/23 15:20', '2023/4/23 17:20')
```

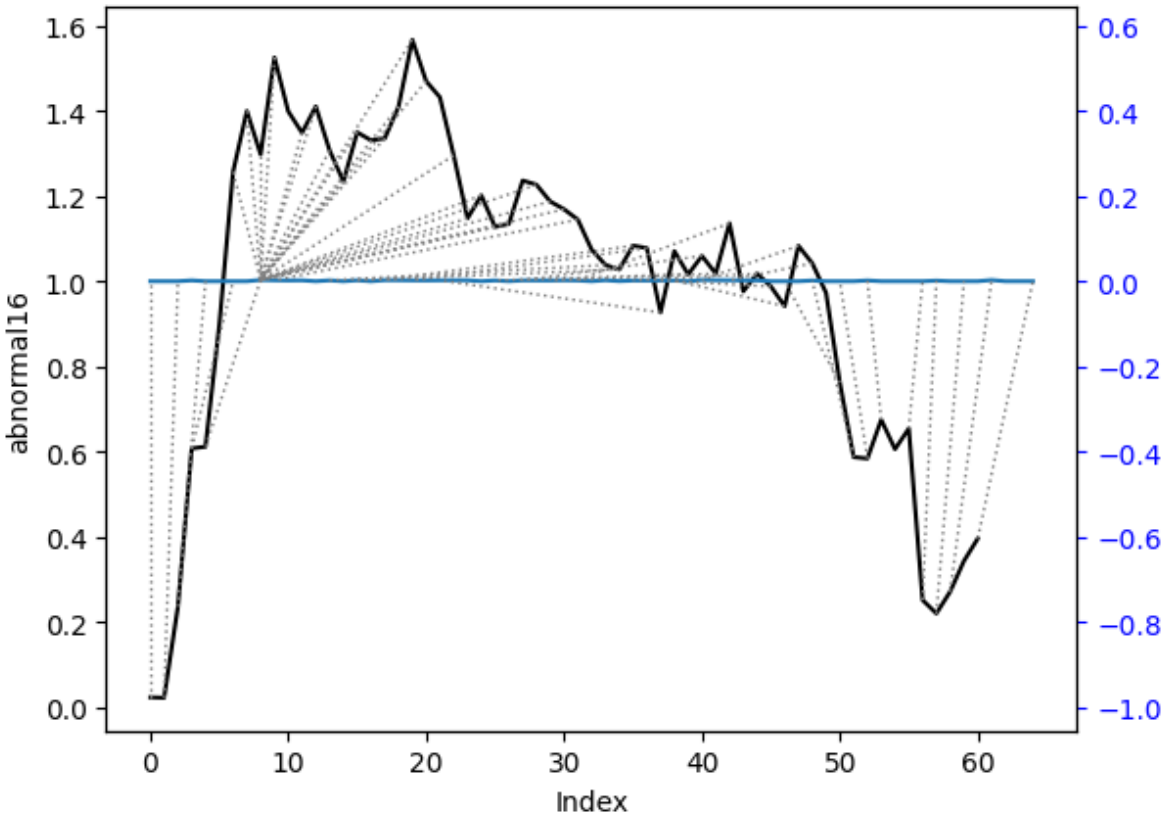
```
相似度排序:
{'node_name': 'abnormal7', 'simi_value': 0.439670667}
{'node_name': 'abnormal8', 'simi_value': 29.780826208000004}
{'node_name': 'abnormal1', 'simi_value': 41.08231588299999}
{'node_name': 'abnormal6', 'simi_value': 49.76684914399999}
{'node_name': 'abnormal13', 'simi_value': 52.82893044699999}
{'node_name': 'abnormal12', 'simi_value': 54.36460776799999}
{'node_name': 'abnormal2', 'simi_value': 58.661397467999976}
{'node_name': 'abnormal15', 'simi_value': 59.404401087999986}
{'node_name': 'abnormal16', 'simi_value': 60.10923679400002}
{'node_name': 'abnormal14', 'simi_value': 374.80343750 2}
{'node_name': 'abnormal17', 'simi_value': 712.1705146799999}
{'node_name': 'abnormal18', 'simi_value': 1079.8687904619997}
{'node_name': 'abnormal10', 'simi_value': 1182.290563162}
{'node_name': 'abnormal11', 'simi_value': 2457.3210061319996}
{'node_name': 'abnormal20', 'simi_value': 5264.824311122}
{'node_name': 'abnormal19', 'simi_value': 9334.395459362004}
{'node_name': 'abnormal9', 'simi_value': 356895.395989462}
{'node_name': 'abnormal3', 'simi_value': 444204.46831347205}
{'node_name': 'abnormal4', 'simi_value': 302892863.49297595}
{'node_name': 'abnormal5', 'simi_value': 488540103.3204277}
```

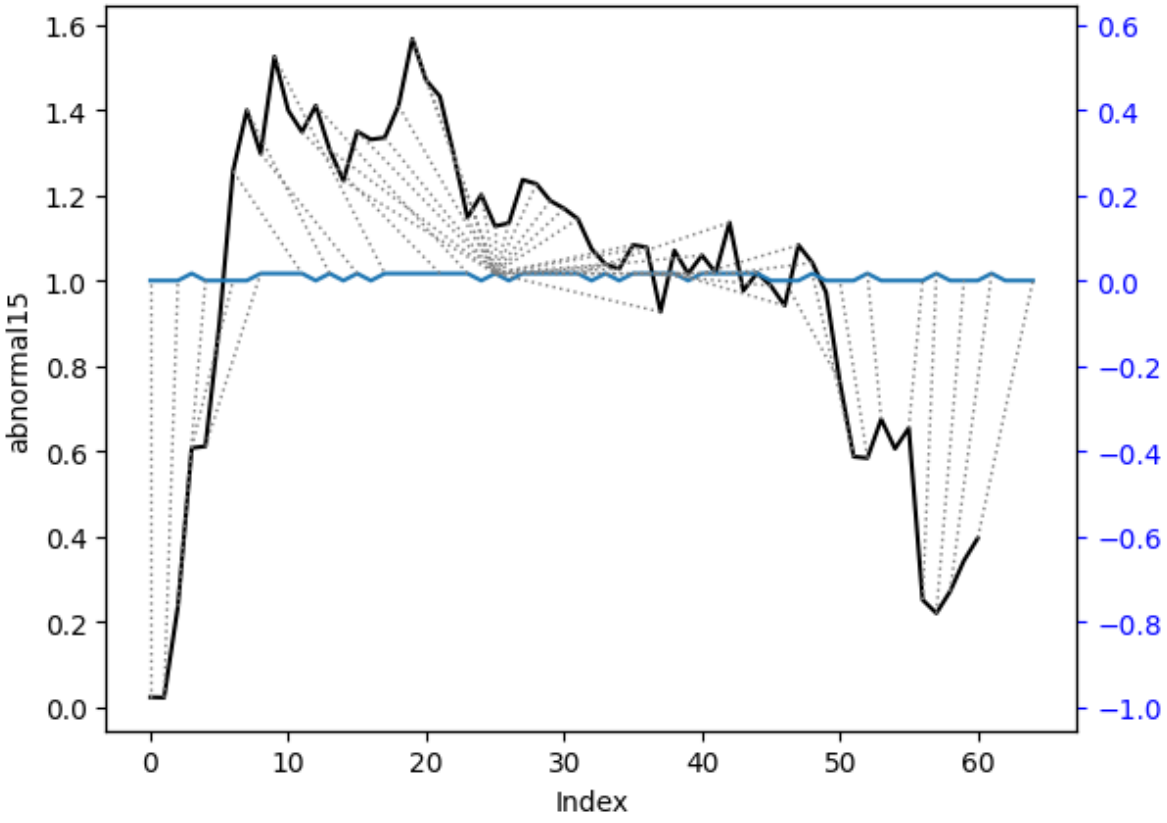

abnormal20
Index
0
10
20
30
40
50
60
0
20
40
60
80
100
120
140

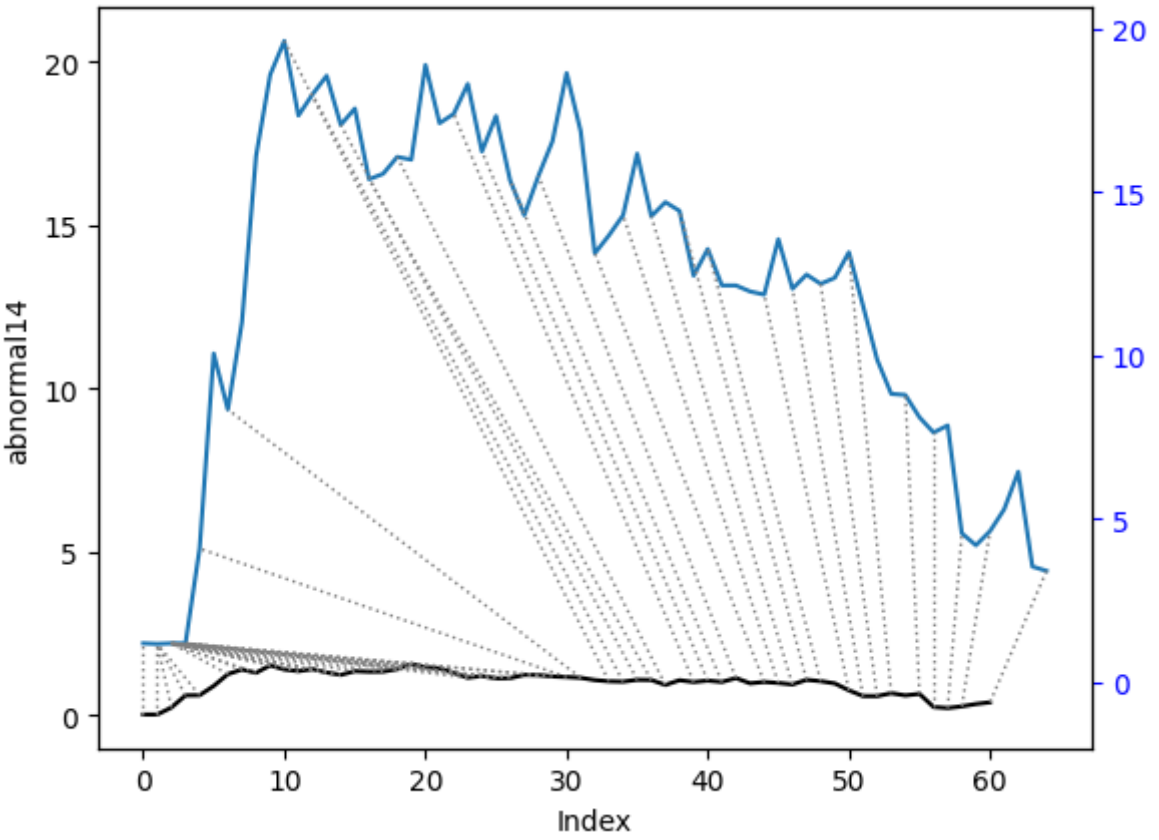

abnormal19
Index
0
10
20
30
40
50
60
0
25
50
75
100
125
150

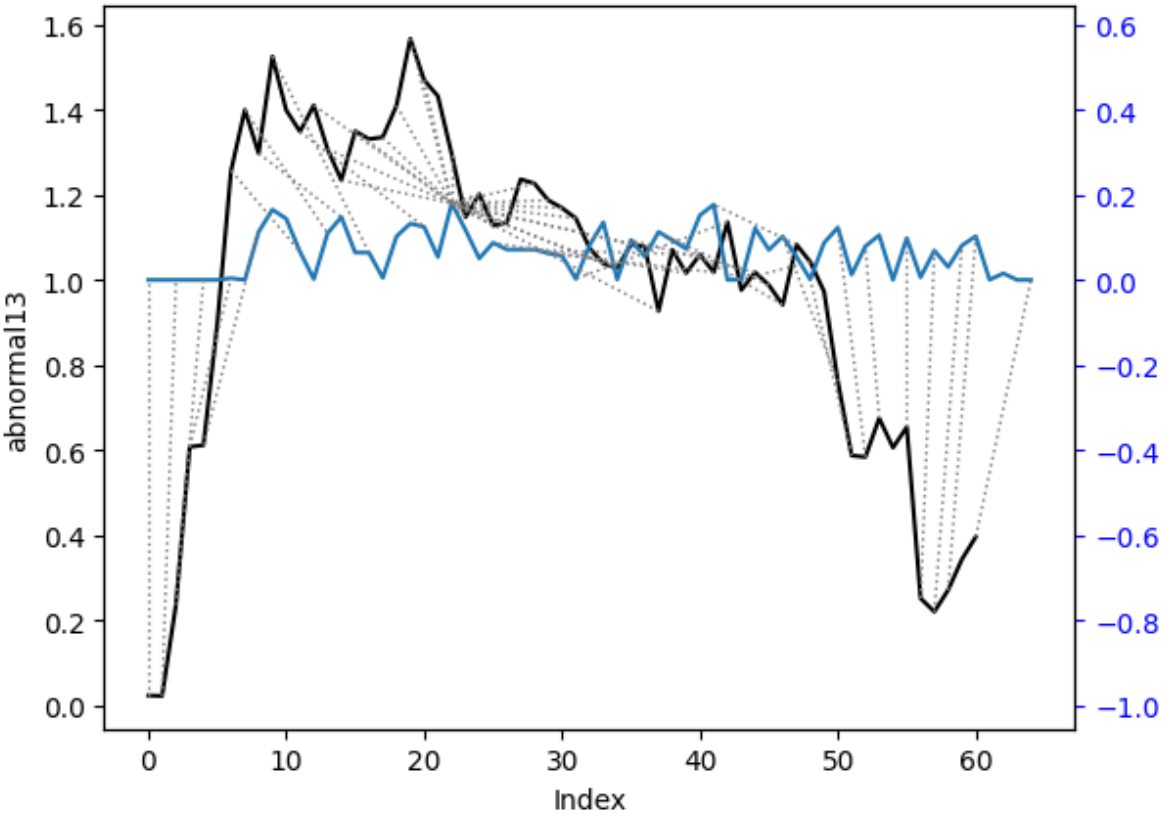

abnormal18
Index
0
10
20
30
40
50
60
0
5
10
15
20
25
30

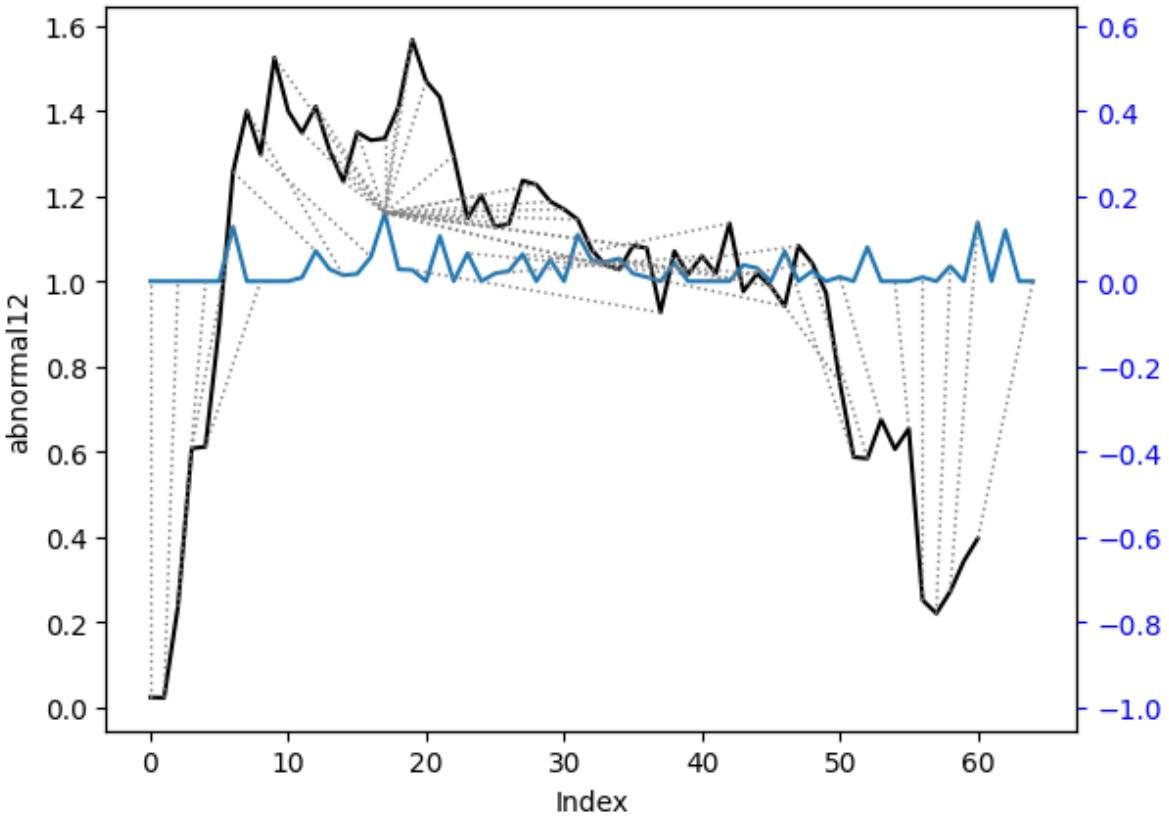

abnormal17
Index
0
10
20
30
40
50
60
0
20
40
60
80

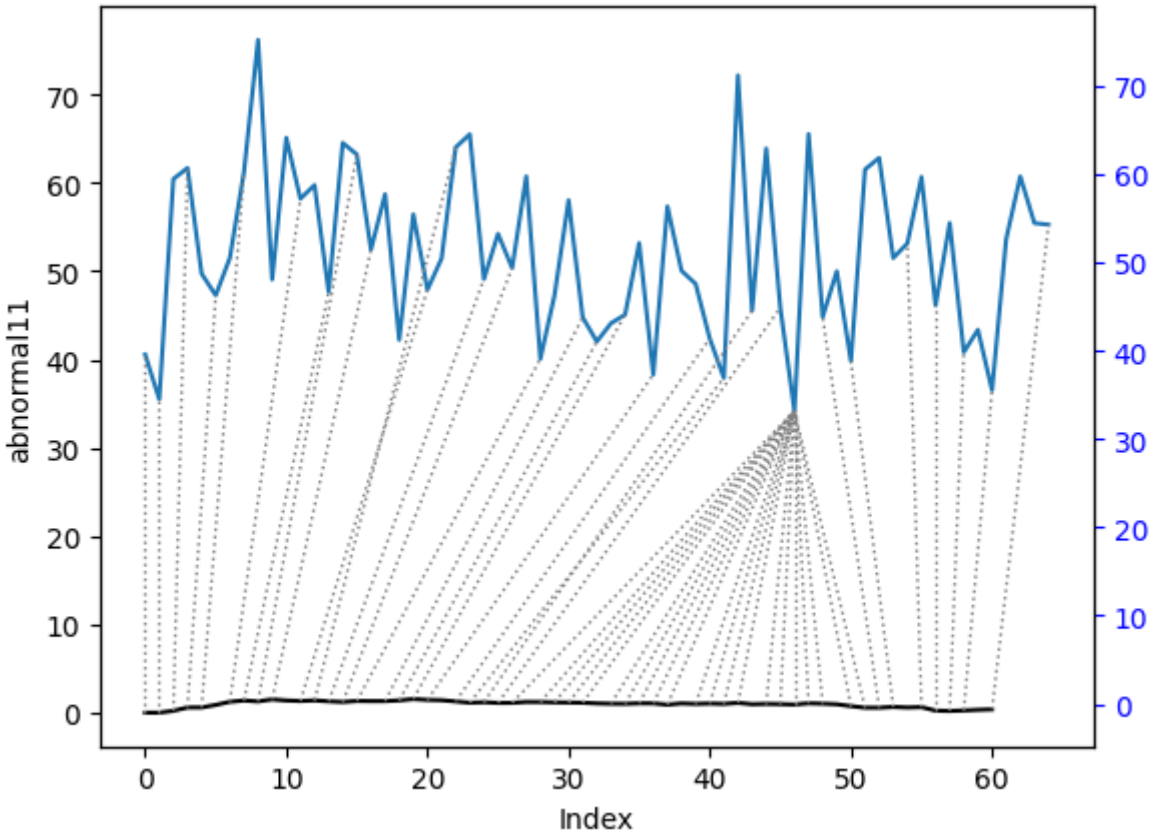

abnormal16
Index
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
10
20
30
40
50
60

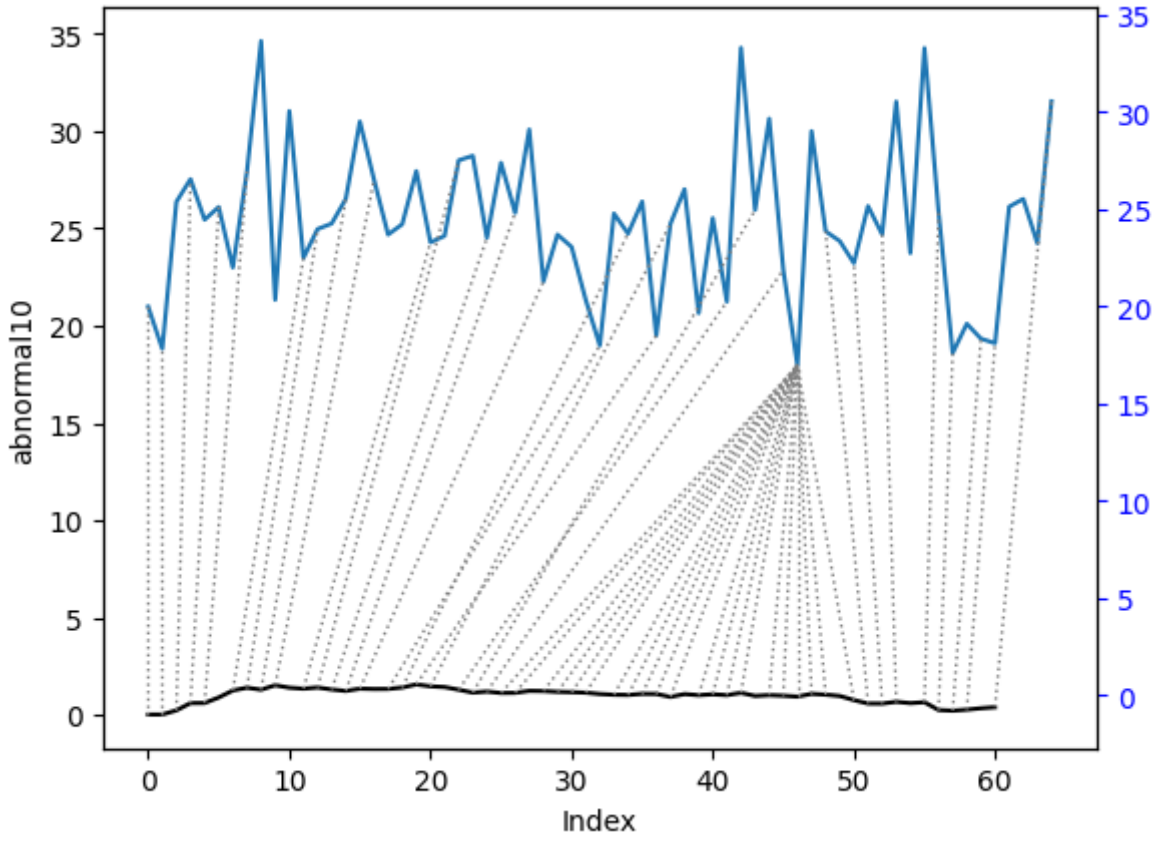

abnormal15
Index
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
10
20
30
40
50
60

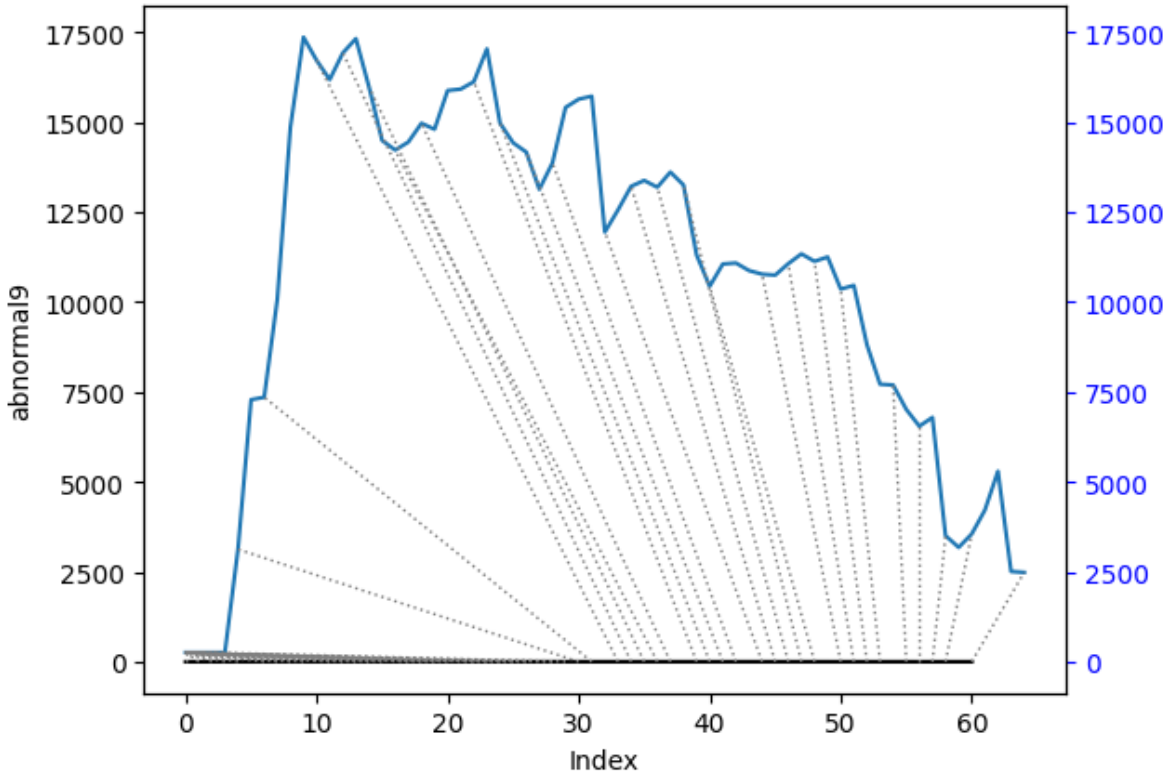

abnormal14
Index
0
5
10
15
20
0
10
20
30
40
50
60

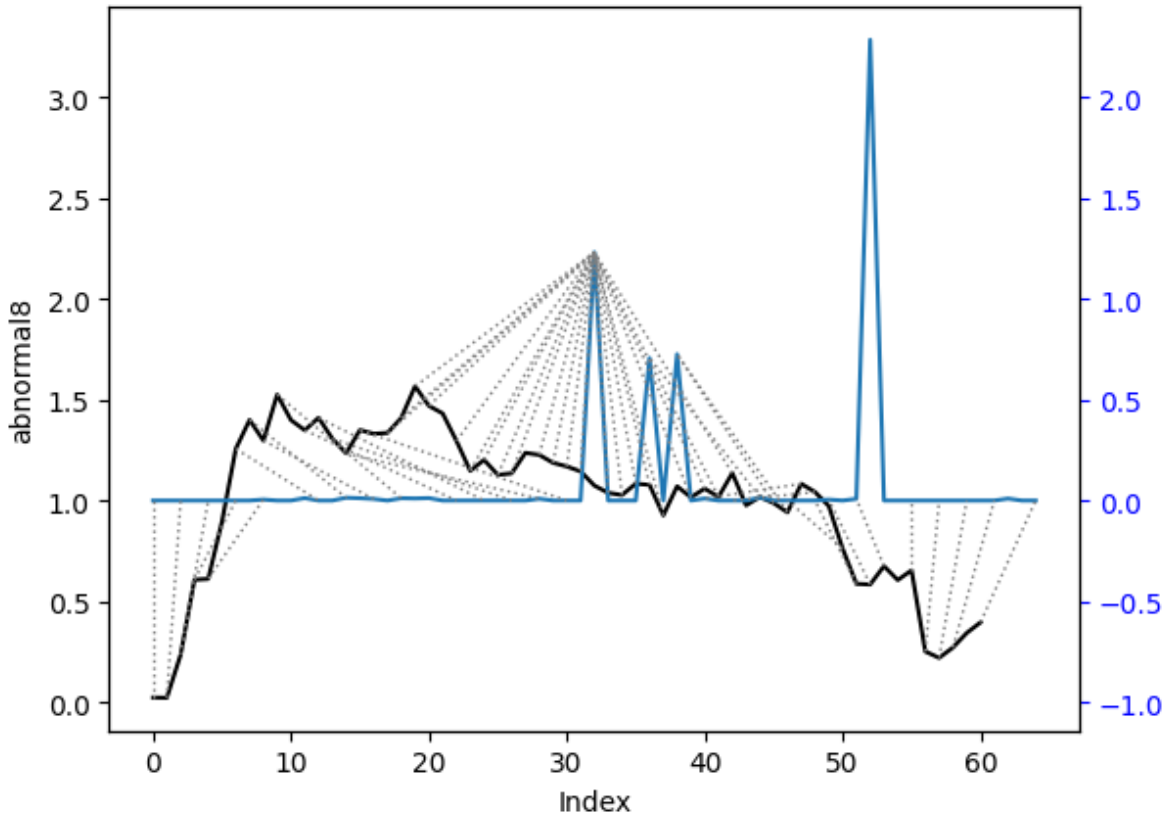

abnormal13
Index
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0
10
20
30
40
50
60

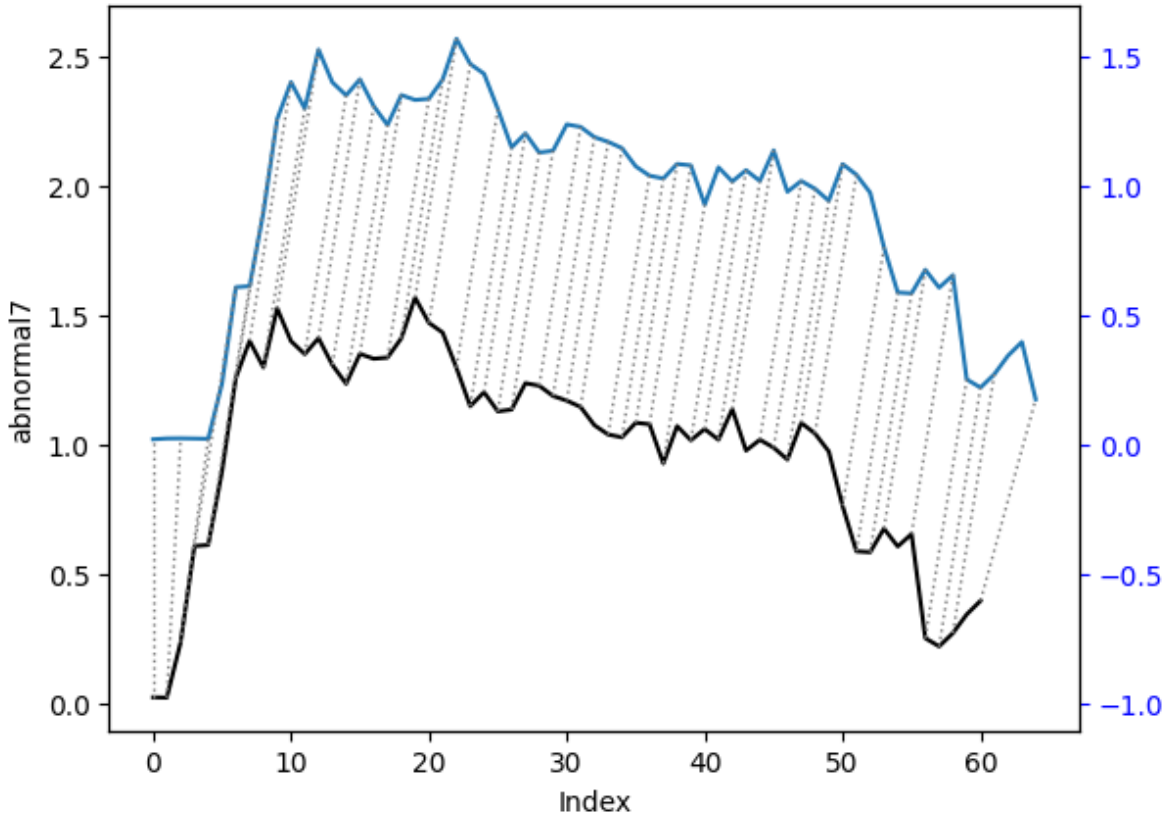

abnormal12
Index
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
10
20
30
40
50
60

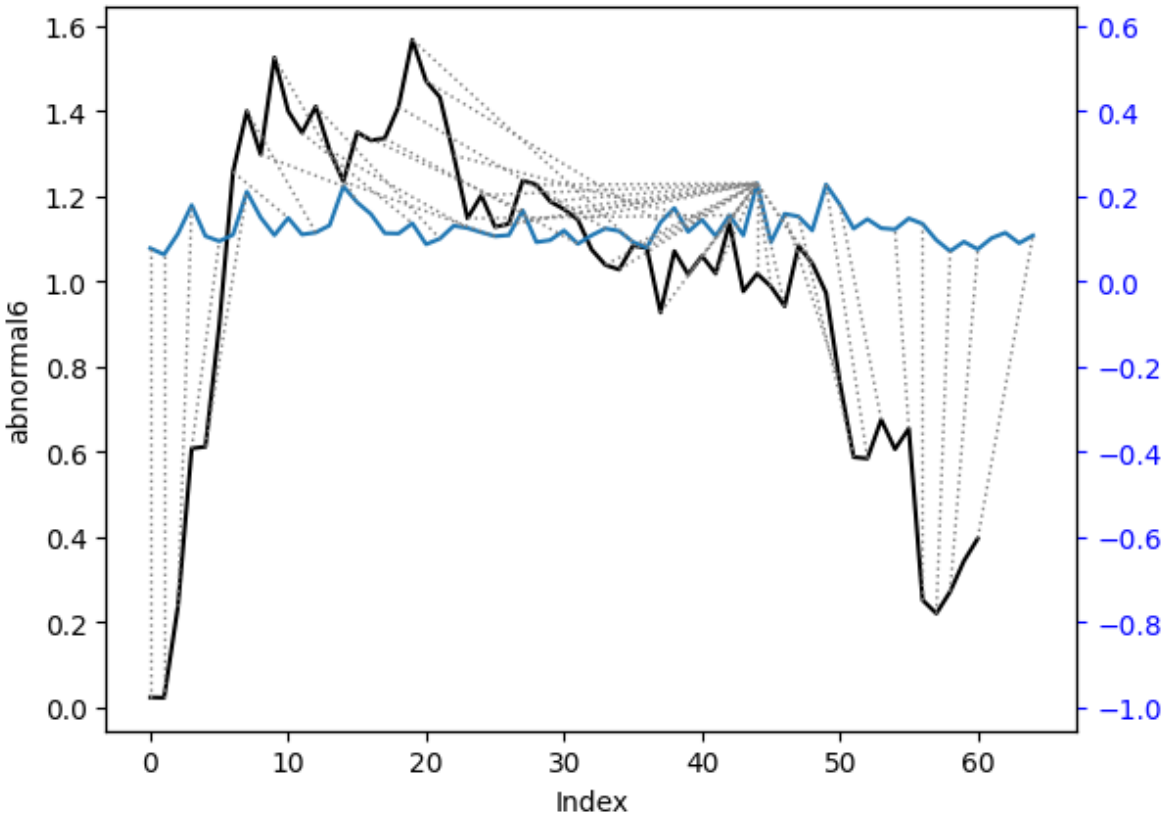

abnormal11
Index
70
60
50
40
30
20
10
0
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60

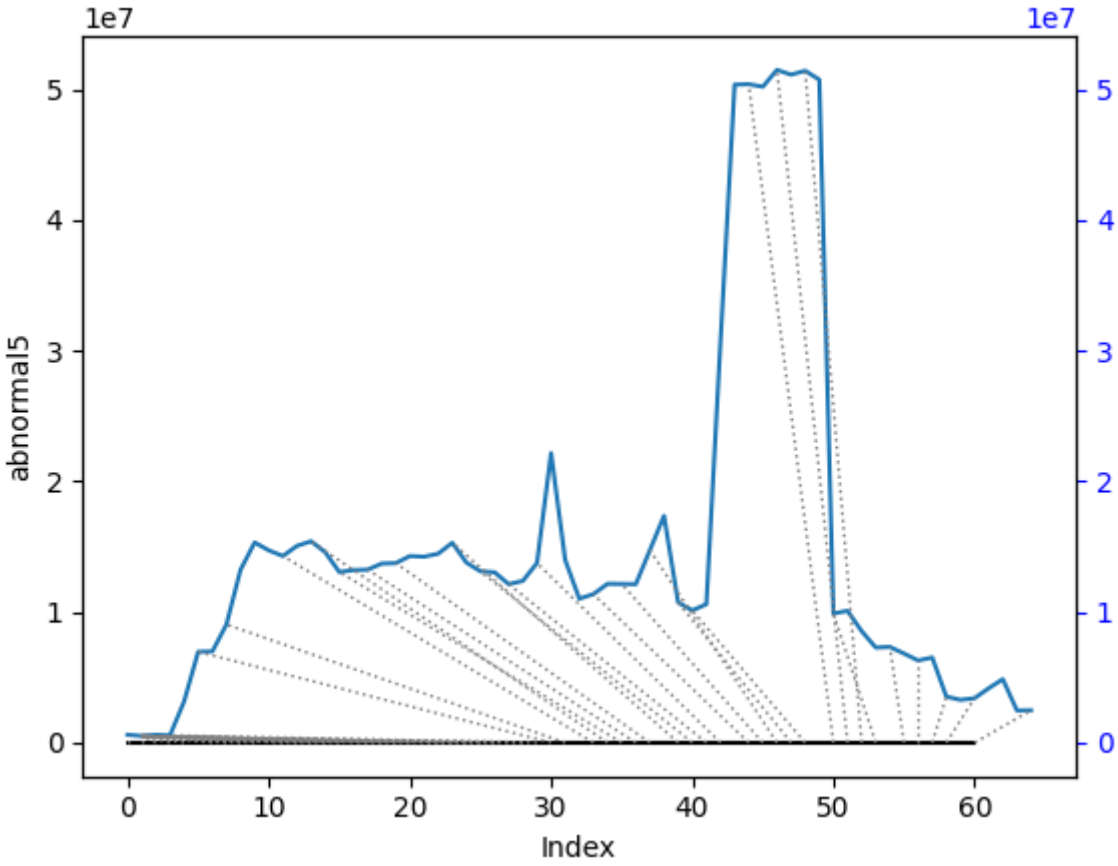

abnormal10
Index
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60

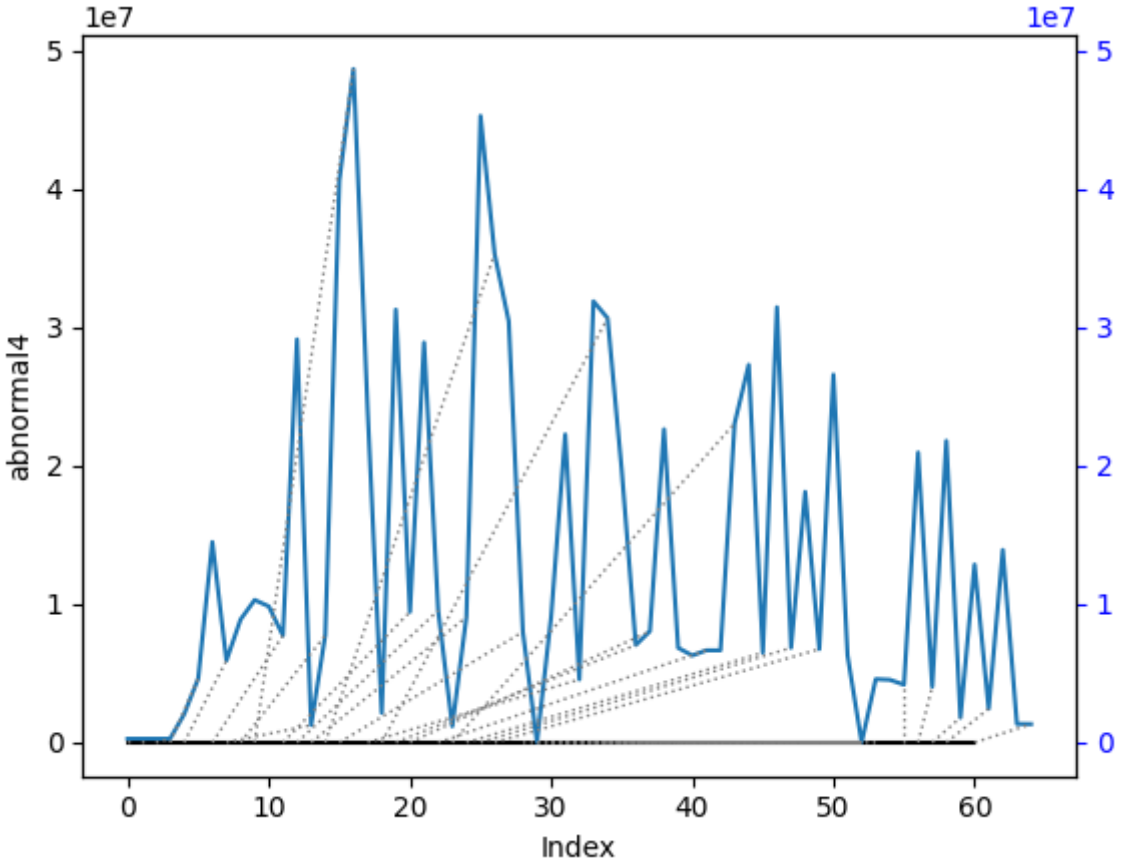

abnormal9
Index
0
2500
5000
7500
10000
12500
15000
17500
0
10
20
30
40
50
60

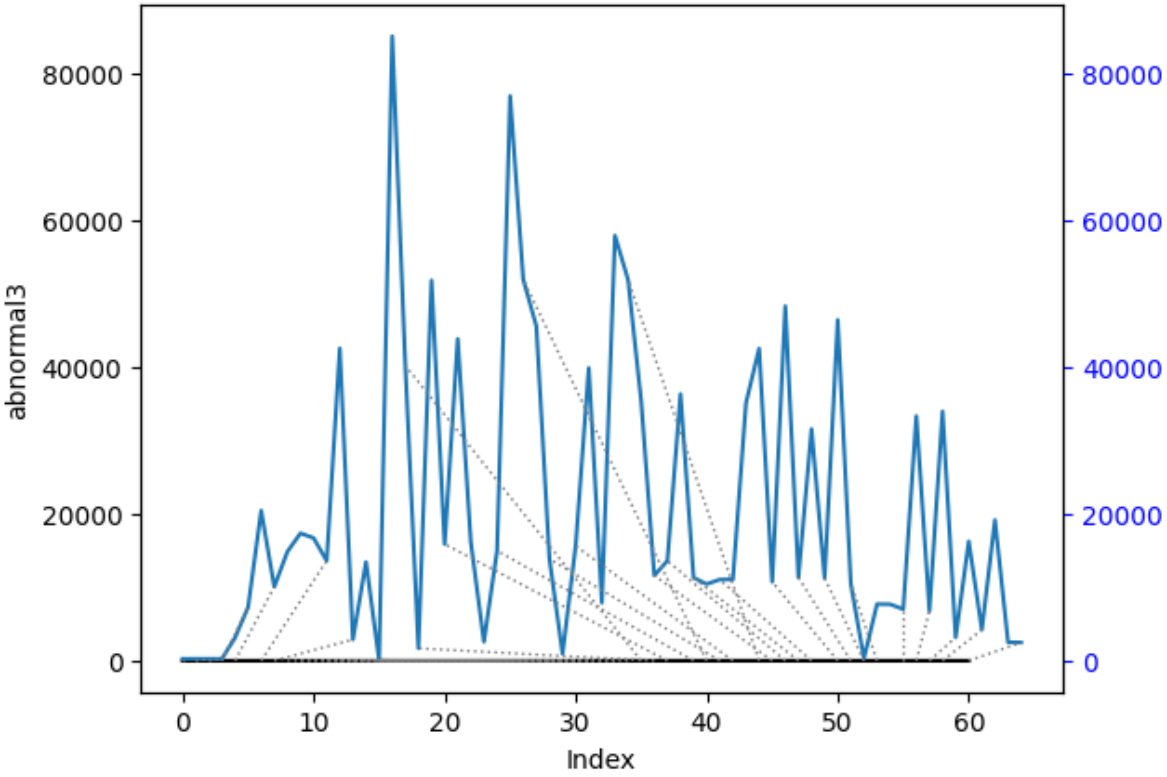

abnormal8
Index
0.0
0.5
1.0
1.5
2.0
2.5
3.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
0
10
20
30
40
50
60

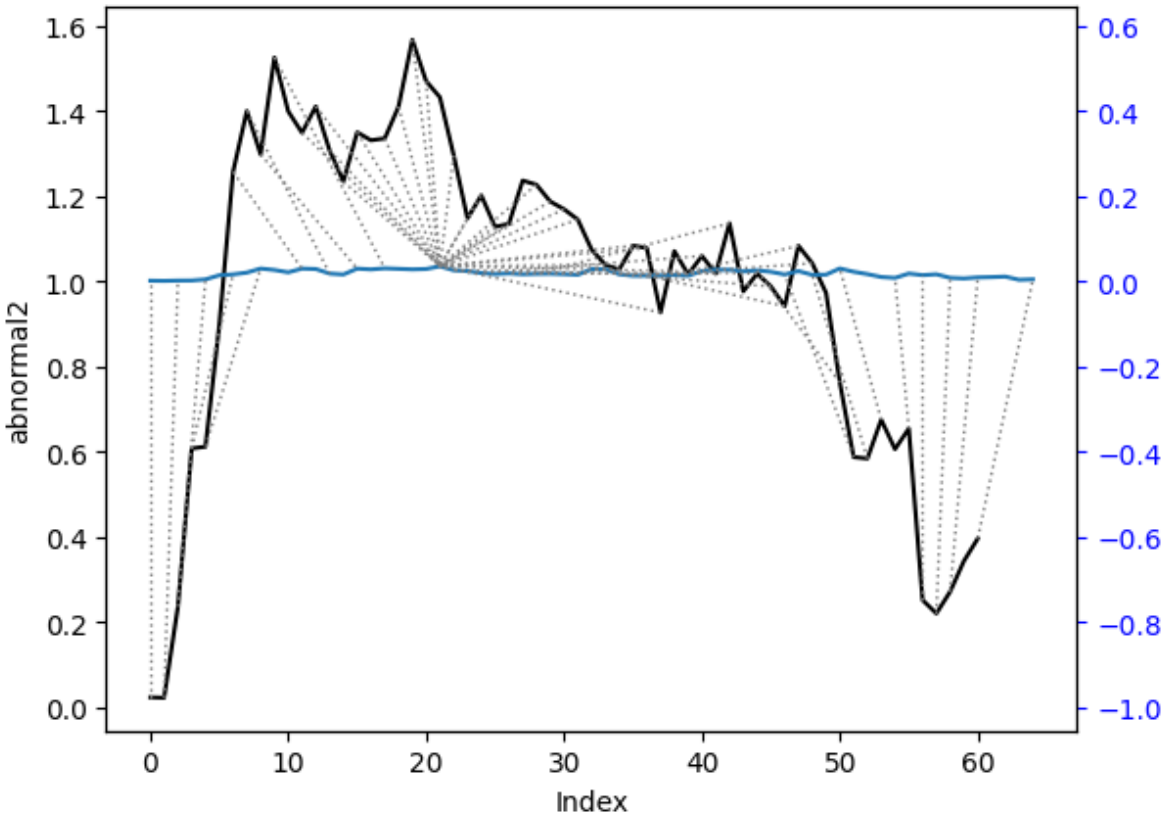

abnormal7
Index
0.0
0.5
1.0
1.5
2.0
2.5
−1.0
−0.5
0.0
0.5
1.0
1.5
0
10
20
30
40
50
60

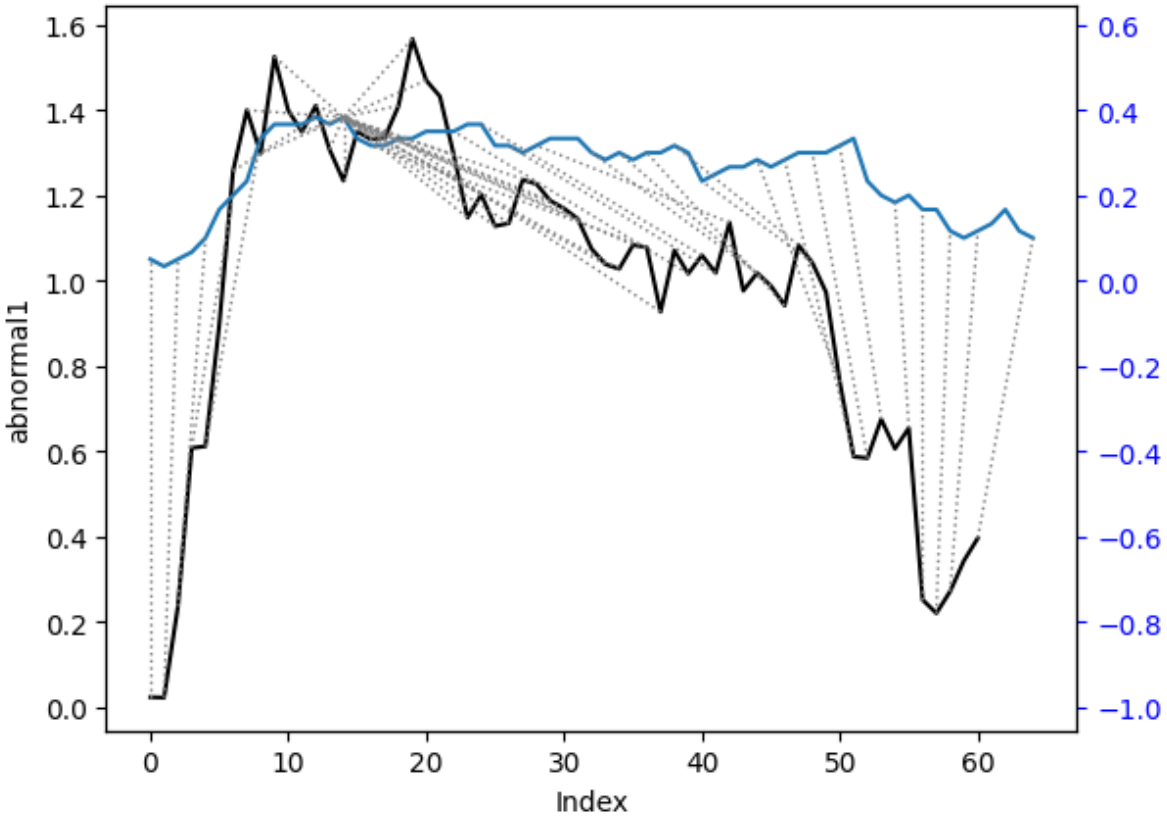

abnormal6
Index
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
10
20
30
40
50
60


abnormal5
Index
1e7
1e7
5
4
3
2
1
0
0
10
20
30
40
50
60


1e7
1e7
abnormal4
Index
0
10
20
30
40
50
60
0
1
2
3
4
5


abnormal3
Index
0
10
20
30
40
50
60
0
20000
40000
60000
80000


abnormal2
Index
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
10
20
30
40
50
60


abnormal1
Index
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
10
20
30
40
50
60