README

TadGAN 알고리즘을 이용한 스마트팜 이상치 탐지 및 생산량 예측

- orion-ml 라이브러리를 이용해서 TadGAN 모델로 이상치를 탐지하고, 탐지된 이상치를 기반으로 생산량을 예측합니다.
- 이상치 탐지는 xinsunadd, xintemp1, xsthum, xco2 4가지 컬럼에 대한 모델이 구현되어있습니다.
- 생산량 예측은 각 이상치의 총합을 feature로 하여 한 작기의 10a당 생산량을 ridge 회귀모델을 이용하여 예측합니다.

데이터

- 경남TP 스마트팜 데이터 이용 2022~2023년 1개 작기
- 딸기(4개 농가), 토마토(5개 농가), 파프리카(1개 농가)
- 분석 DB

모델

1. 이상치 탐지 모델

- orion-ml 라이브러리의 Tadgan 모델을 사용하여 이상치 탐지
- 시간 집계는 1시간 (3600초)을 기준으로 집계
- 5 epoch 학습

데이터셋

- 시계열 데이터셋
- 필수 컬럼 중 xintemp는 xintemp1만 사용 후 나머지는 제, xco2set은 분석에서 제외함
  1. xintemp2의 경우 xintemp1과 매우 높은 상관관계(0.99)가 보이며, 데이터의 스케일이 같으므로 모델링에서 제외함 (Correlation Heatmap.jpeg 참고)
  2. xco2set의 경우 1/4이상이 0.0으로 기록되어 있으며, 일부 농가에서는 3/4 가량이 0.0이므로 결측치가 너무 많다고 판단되어 모델링에서 제외 (Boxplot.jpeg 참고)

- datetime과 탐지하고자하는 환경데이터로 구성된 데이터셋을 사용

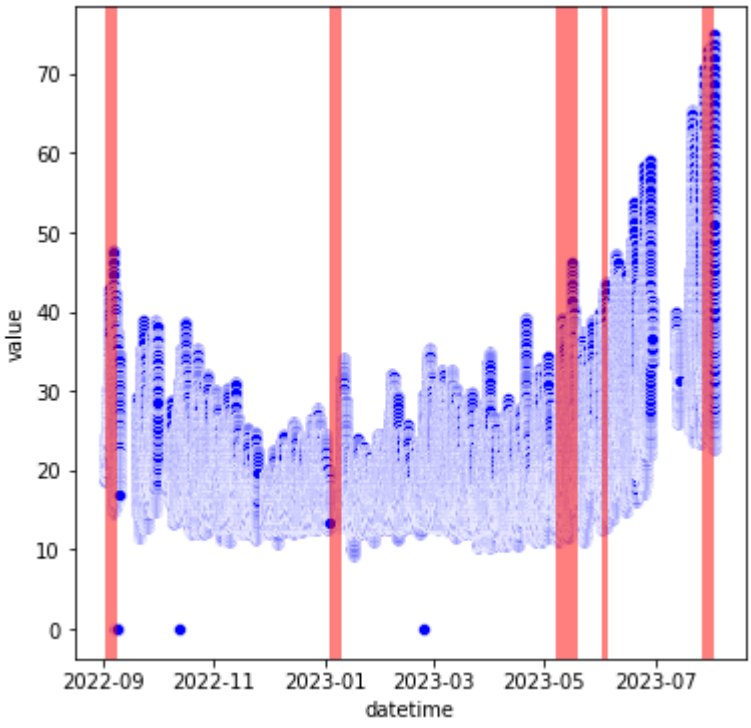
	xdatetime	xinsunadd	xintemp1	xsthum	xco2	zone
0	2022-09-02 10:50:00	0.0	24.5	23.0	336.0	mysb6_1
1	2022-09-02 10:51:00	0.0	24.5	23.0	334.0	mysb6_1
2	2022-09-02 10:52:00	0.0	24.4	22.9	335.0	mysb6_1
3	2022-09-02 10:53:00	0.0	24.4	22.9	332.0	mysb6_1
4	2022-09-02 10:54:00	0.0	24.3	22.8	332.0	mysb6_1
...	...	...	...	...	...	...
1552885	2023-08-02 11:00:00	2172.0	39.6	53.1	376.0	mysb4_4
1552886	2023-08-02 11:00:00	2101.0	41.1	58.6	540.0	mysb6_5
1552887	2023-08-02 11:00:00	2163.0	39.0	52.4	324.0	mysb6_6
1552888	2023-08-02 11:01:00	2231.0	35.2	42.6	312.0	mysb6_1
1552889	2023-08-02 11:01:00	2106.0	41.2	58.6	539.0	mysb6_5

Returns

- Tadgan 모델은 point anomaly를 탐지하는 것이 아니라, context anomaly를 탐지하므로, 기간과 severity가 출력
- 출력값 형태는 아래와 같음

	start	end	severity
0	2023-02-23 00:00:00	2023-02-27 14:00:00	1.467459
1	2023-03-09 16:00:00	2023-03-12 09:00:00	0.175722
2	2023-03-26 23:00:00	2023-03-31 13:00:00	1.469136

- TadGAN이 탐지한 이상치 예시
  - 파란 점 : 관측 값
  - 빨간 색칠 부분 : 이상치로 탐지된 기간



2. 생산량 예측 모델

- 선형회귀 모델 사용 (Ridge 회귀)
- 학습 데이터셋의 부족 문제로 일반화 성능이 부족하여 alpha값을 많이 주어 예측값이 민감하게 반응하지 않음

학습 데이터셋

- 농가별 총 탐지된 이상치의 총 합을 feature로, 10a당 생산량 중 medain으로부터의 편차를 target으로 하여 데이터셋 생성

	xco2	xinsunadd	xintemp1	xsthum	yield_output	output_dev
mysb6_6	0.296633	3.128465	1.057159	1.392112	1759.67	312.395
mysb6_1	0.000000	0.260533	1.217159	1.610267	1090.28	-356.995
mysb4_4	0.258910	1.731777	0.343589	0.785810	1671.79	224.515
mysb6_5	0.033354	2.677468	0.810482	2.041166	1267.36	-179.915

-

학습 프로세스

- 농가별 이상치의 총 합을 feature로 함
- scaling : RobustScaler를 이용하여 median과 IQR을 이용해 robust한 스케일링을 진행
- target : scaled된 median으로부터의 편차를 target으로 하여 학습을 진행
- **paprica의 경우 생산량데이터가 1개로, 회귀식 생성이 불가능해 예측이 불가능**(predict 함수 사용 시 저번 작기 생산량 값을 출력)

Returns

- 2차원 np.array형태안에 값이 들어있는 형태

```
Out[65]: array([[5705.79]])
```

사용법 예시

```
if __name__ == '__main__':
        farm = Tadgan(str(sys.argv[1]))
        farm.load_data()
        farm.load_model()
        farm.detect('xinsunadd')
        farm.predict()
```

- main 함수는 사용 형태에 맞게 변경해서 사용

In [68]:
```
mysb2_1 = Tadgan('mysb2_1')
```

In [69]:
```
mysb2_1.load_data()
```

Out[69]:

	xdatetime	xinsunadd	xintemp1	xsthum	xco2	zone
0	2022-09-02 10:50:00	0.0	24.1	22.5	381.0	mysb2_1
1	2022-09-02 10:51:00	0.0	24.0	22.4	380.0	mysb2_1
2	2022-09-02 10:52:00	0.0	24.0	22.4	378.0	mysb2_1
3	2022-09-02 10:53:00	0.0	23.8	22.1	377.0	mysb2_1
4	2022-09-02 10:54:00	0.0	23.8	22.1	379.0	mysb2_1
...	...	...	...	...	...	...
267619	2023-04-05 13:53:00	4133.0	19.2	16.7	492.0	mysb2_1
267620	2023-04-05 13:54:00	4141.0	19.2	16.7	491.0	mysb2_1
267621	2023-04-05 13:55:00	4150.0	19.2	16.7	487.0	mysb2_1
267622	2023-04-05 13:56:00	4159.0	19.2	16.7	489.0	mysb2_1
267623	2023-04-05 13:57:00	4168.0	19.1	16.6	489.0	mysb2_1

267624 rows × 6 columns

In [70]:
```
mysb2_1.load_model()
```

...

In [71]:
```
mysb2_1.detect('xinsunadd', '2023-02-07')
```

```
=====    xinsunadd detected     =====
```

Out[71]:

	start	end	severity
0	2023-02-23 00:00:00	2023-02-27 14:00:00	1.467459
1	2023-03-09 16:00:00	2023-03-12 09:00:00	0.175722
2	2023-03-26 23:00:00	2023-03-31 13:00:00	1.469136

In [64]:
```
mysb2_1.detect('xinsunadd', ['2023-02-07', '2023-03-07'])
```

```
=====    xinsunadd detected     =====
```

Out[64]:

	start	end	severity
0	2023-02-22 13:00:00	2023-02-27 12:00:00	0.312055

In [65]:
```
mysb2_1.predict()
```

```
=====    xinsunadd detected     =====
=====    xintemp1 detected      =====
=====    xsthum detected        =====
=====    xco2 detected  =====
[[5705.79]]
```

Out[65]:
```
array([[5705.79]])
```

Release Note

ver1.1 : 2023. 8. 31.