설비별 실시간 역률 예측을 통한 범용설비의 설비이상 진단 및 역률 개선을 위한 원인분석

2021 빅데이터 경진대회

산업정보시스템전공 강채원 이현진 임수진

목차

01 프로젝트 주제 및 배경

* 관련 용어 설명

상전압	한 상에 걸리는 전압
선간전압	두 상간의 전압의 차
유효전력	부하에서 실제로 유효한 일로 바뀌는 전력 (저항에서 열로 소비하는 전력이자 실제로 사용하는 전력)
무효전력	연결된 부하에서 소비되지 않는 전력 (실제로 아무 일도 하지 않고 열 소비도 하지 않는 전력)
고조파	기본 주파수에 대해 2배, 3배, 4배와 같이 정수의 배에 해당하는 물리적 전기량으로써, 전력망의 고조파 주파수는 전력 품질 문제, 전력 시스템의 고조파는 장비 및 도체의 가열 증가, 가변 속도 드라이브의 오작동 및 모터의 토크 맥동을 초래
역률	전기를 사용할 때 실제 일을 하는 유효 전력에 비해 일은 하지 않고 소모만 되는 무효전력이 얼마나 적게 차지하는가의 비율 (전류가 단위시간에 하는 일의 비율)

01 프로젝트 주제 및 배경

* 문제 정의

현재 품질관리 시스템

- 한전 측에서 30분 단위로 측정한 전체 사용 설비의 역률 평균을 통한 역률 요금 청구

- 역률이 떨어진 설비를 바로 알려주는 시스템의 부재

■ 제 43 조 [역률에 따른 요금의 추가 또는 감액]

① 역률에 따른 요금의 추가 또는 감액 대상 고객은 제38조(전력량계 등의 설치기준)에 따라 무효전력을 계량할 수 있는 전력량계가 설치된 고객으로서 다음 각 호에서 정한 고객으로 합니다.
1. 저압으로 전기를 공급받는 계약전력 20kW 이상의 일반용전력, 산업용전력, 농사용전력, 임시전력
2. 고압이상의 전압으로 전기를 공급받는 일반용전력, 교육용전력, 산업용전력, 농사용전력, 임시전력

② 역률에 따른 요금의 추가 또는 감액은 다음 각 호와 같이 시간대별로 구분하여 산정합니다.
1. 09시부터 23시까지의 역률에 따른 요금의 추가 또는 감액
가. 지상역률(遲相力率)에 대하여 적용하며, 평균역률이 90%에 미달하는 경우에는 미달하는 역률 60%까지 매 1%당 기본요금의 0.2%를 추가하고, 평균역률이 90%를 초과하는 경우에는 역률 95%까지 초과하는 매 1%당 기본요금의 0.2%를 감액합니다.
나. "가"의 평균역률은 제42조(역률의 계산) 제2항의 본문에 따라 계산된 30분단위의 역률을 1개월간 평균하여 계산합니다. 다만, 30분 단위의 역률이 지상역률 60%에 미달하는 경우 역률 60%로, 지상역률 95%를 초과하는 경우 역률 95%로 간주하여 1개월간 평균역률을 계산합니다.
다. "나"에도 불구하고, 제42조(역률의 계산) 제2항의 단서에 해당하는 고객의 평균역률은 전력량계에 1개월간 누적된 계량값으로 역률을 계산합니다.
2. 23시부터 다음 날 09시까지의 역률에 대한 요금의 추가
가. 진상역률(進相力率)에 대하여 적용하며, 평균역률이 95%에 미달하는 경우에 미달하는 매 1%당 기본요금의 0.2%를 추가합니다.
나. "가"의 평균역률은 제42조(역률의 계산) 제2항의 본문에 따라 계산된 30분단위의 역률을 1개월간 평균하여 계산합니다. 다만, 30분 단위의 역률이 진상역률 60%에 미달하는 경우에는 역률 60%로, 지상역률인 경우에는 역률 100%로 간주하여 1개월간 평균역률을 계산합니다.
다. "나"에도 불구하고, 제42조(역률의 계산) 제2항의 단서에 해당하는 고객은 진상역률 요금을 적용하지 않습니다.

③ 해당월에 지상역률 또는 진상역률의 추가요금이 발생한 경우 첫 번째 달에는 추가요금의 청구를 예고하고 두 번째 달부터 추가요금을 청구합니다.

분석 목표

- **<u>설비별</u> 실시간 역률 예측**

전력 손실 경감

전기 요금 경감

설비 용량의 여유 증가

갑작스러운 가동중단으로 인한 막대한 손해 방지

01 프로젝트 주제 및 배경

* 문제 정의

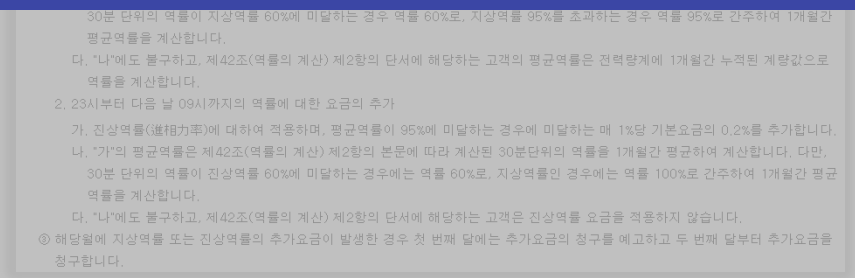
현재 품질관리 시스템

- 역률이 떨어지면 바로 알려주는 시스템의 부재

분석 목표

- 설비별 실시간 역률 예측

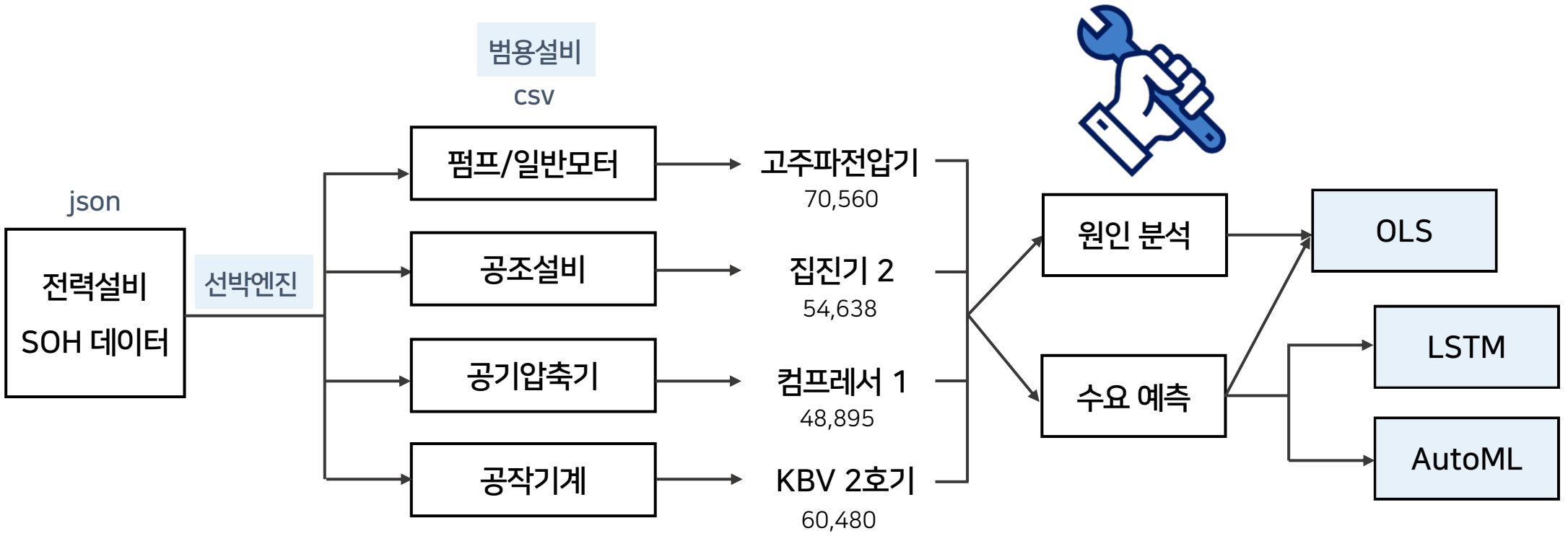
설비별 실시간 역률 예측을 통한 범용설비의 설비이상 진단 및 역률 개선을 위한 원인분석


30분 단위의 역률이 지상역률 60%에 미달하는 경우 역률 60%로, 지상역률 95%를 초과하는 경우 역률 95%로 간주하여 1개월간 평균역률을 계산합니다.
다. "나"에도 불구하고, 제42조(역률의 계산) 제2항의 단서에 해당하는 고객의 평균역률은 전력량계에 1개월간 누적된 계량값으로 역률을 계산합니다.
2. 23시부터 다음 날 09시까지의 역률에 대한 요금의 추가
가. 진상역률(進相力率)에 대하여 적용하며, 평균역률이 95%에 미달하는 경우에 미달하는 매 1%당 기본요금의 0.2%를 추가합니다.
나. "가"의 평균역률은 제42조(역률의 계산) 제2항의 본문에 따라 계산된 30분단위의 역률을 1개월간 평균하여 계산합니다. 다만, 30분 단위의 역률이 진상역률 60%에 미달하는 경우에는 역률 60%로, 지상역률인 경우에는 역률 100%로 간주하여 1개월간 평균역률을 계산합니다.
다. "나"에도 불구하고, 제42조(역률의 계산) 제2항의 단서에 해당하는 고객은 진상역률 요금을 적용하지 않습니다.
③ 해당월에 지상역률 또는 진상역률의 추가요금이 발생한 경우 첫 번째 달에는 추가요금의 청구를 예고하고 두 번째 달부터 추가요금을 청구합니다.

갑작스러운 가동중단으로 인한 막대한 손해 방지

02 전처리 및 EDA

* 분석 프로세스



02 전처리 및 EDA

* 설비 선정

- 주의, 경고 등 기계에 이상이 생기면
  큰 문제가 발생하는 '범용 설비'를 대상으로 추출

분류	계측 대상 설비
에너지 다소비 설비 (6종)	보일러, 가열로, 압출기, 사출기, 주조기, 열처리
범용 설비 (4종)	펌프/일반모터, 공조설비, 공기압축기, 공작기계

* 변수 추출

- 데이터 중분류 중, 평균값에 해당하는 데이터 추출

<표. 설비별 전력 에너지 및 전력 품질 데이터 요약표>

	데이터 중분류	데이터 소분류	단위	내 용	유 형
1	누적 전력량	합계	kWh	3 상 누적 전력량의 합	시계열/Value(float)
2	상전압	R상	V	R상 전압	시계열/Value(float)
3		S상	V	S상 전압	시계열/Value(float)
4		T상	V	T상 전압	시계열/Value(float)
5		평균	V	3상 전압의 평균	시계열/Value(float)
6	선간전압	RS	V	R상 전압	시계열/Value(float)
7		ST	V	S상 전압	시계열/Value(float)
8		TS	V	T상 전압	시계열/Value(float)
9		평균	V	3상 전압의 평균	시계열/Value(float)
10	전류	R상	A	R상 전류	시계열/Value(float)
11		S상	A	S상 전류	시계열/Value(float)
12		T상	A	T상 전류	시계열/Value(float)
13		평균	A	3상 전압의 평균	시계열/Value(float)
14	유효 전력	R상	kW	R상 유효전력 순시값	시계열/Value(float)
15		S상	kW	S상 유효전력 순시값	시계열/Value(float)
16		T상	kW	T상 유효전력 순시값	시계열/Value(float)
17		평균	kW	3상 유효전력 순시값의 평균	시계열/Value(float)
18	무효 전력	R상	kVar	R상 무효전력 순시값	시계열/Value(float)
19		S상	kVar	S상 무효전력 순시값	시계열/Value(float)
20		T상	kVar	T상 무효전력 순시값	시계열/Value(float)
21		평균	kVar	3상 유효전력 순시값의 평균	시계열/Value(float)
22	역률	R상	%	R상 역률(진상,지상 부호 포함)	시계열/Value(float)
23		S상	%	S상 역률(진상,지상 부호 포함)	시계열/Value(float)
24		T상	%	T상 역률(진상,지상 부호 포함)	시계열/Value(float)
25		평균	%	3상 역률 평균	시계열/Value(float)
26	전압 고조파	R상	%	R상 전압 총고조파 왜율	시계열/Value(float)
27		S상	%	S상 전압 총고조파 왜율	시계열/Value(float)
28		T상	%	T상 전압 총고조파 왜율	시계열/Value(float)
29		평균	%	3상 전압 총고조파 왜율 평균	시계열/Value(float)
30	전류 고조파	R상	%	R상 전류 총고조파 왜율	시계열/Value(float)
31		S상	%	S상 전류 총고조파 왜율	시계열/Value(float)
32		T상	%	T상 전류 총고조파 왜율	시계열/Value(float)
33		평균	%	3상 전류 총고조파 왜율 평균	시계열/Value(float)
34	주파수	3상	Hz	전압 주파수	시계열/Value(float)
35	온도	3상	℃	계측부하의 온도	시계열/Value(float)

02 전처리 및 EDA

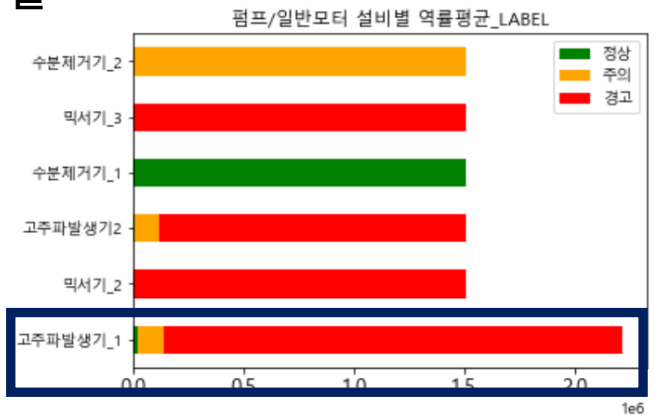
*** 선박엔진 추출**

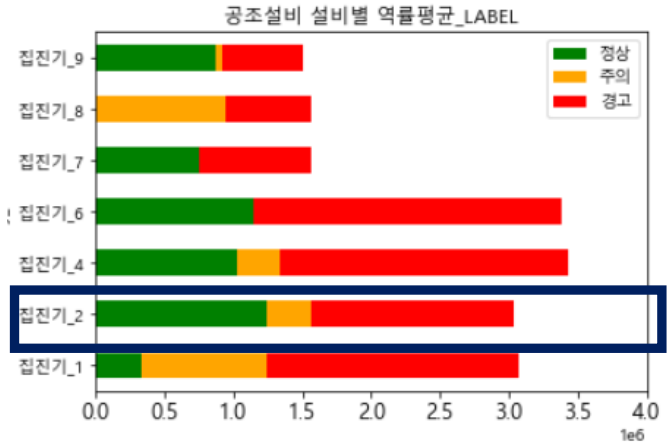
- 설비 4개별로 MAJOR_PRODUCT가 '선박엔진'인 데이터 추출
- 회사 1곳에 한정해 데이터 추출 (시계열 매핑)

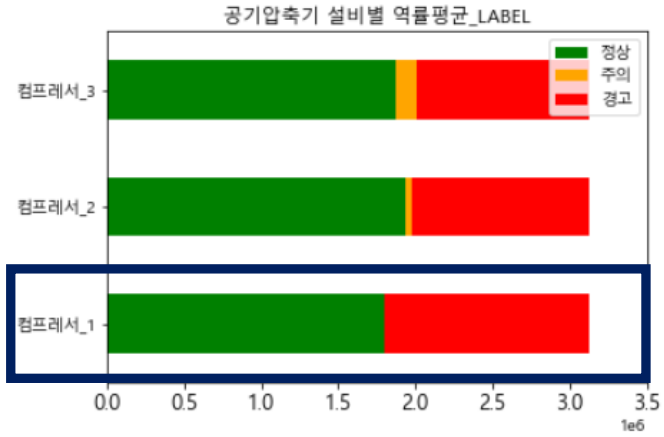
	펌프/일반모터	공조설비	공기압축기	공작기계
선박엔진	9,777,600	17,544,435	17,587,373	9,979,200
자동차부품	7,056,000	9,634,450	13,852,529	73,778,941
화학섬유직물	6,955,200	3,477,600	-	-
기계 부품 열처리	-	2,721,600	1,360,800	-
중장비 부품	-	1,612,800	1,612,800	13,503,840
열처리	-	1,509,760	-	-
원단	4,335,555	1,411,200	3,972,850	-
금속 열처리	-	-	3,678,885	-
철구조물	-	-	10,229,590	-
섬유	3,512,567	-	8,104,611	-
농기계 부품	178,640	-	504,000	-
자동차 차제용 부품	-	-	5,443,200	18,084,500
플라스틱 착색제	-	-	4,978,080	-
볼트,너트,열처리	-	-	3,124,800	14,963,865
치과용 의료기구	-	-	1,667,750	12,756,450
냉연철강재 코일	2,327,080	-	1,260,000	-
사출금형	-	-	-	26,661,040
플라스틱 사출	-	-	-	5,285,770
자동차 내장재	-	-	-	5,265,098
전자부품	-	-	-	2,016,000
기계가구	-	-	-	856,800
맨홀뚜껑	-	-	-	1,562,400

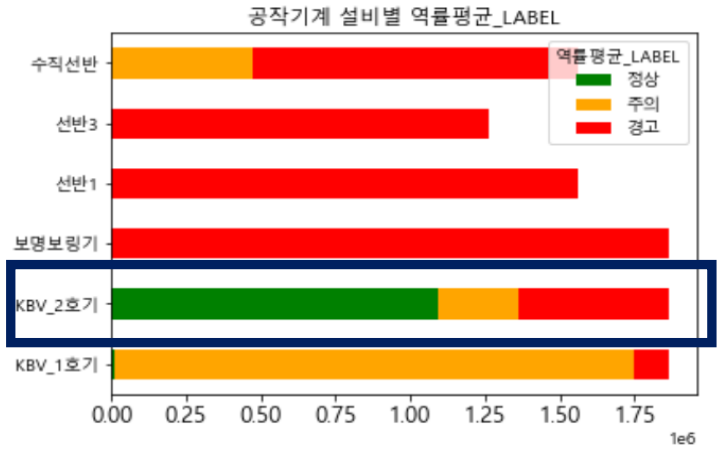
02 전처리 및 EDA

* 설비 당 기계 1개 추출









설비 기계별 역률평균 LABEL 분포 시각화를 통해 각 설비를 대표할 수 있는 기계 선정

선정 기준 : LABEL이 정상, 주의, 경고 모두 고르게 분포하고 있는지 여부

02 전처리 및 EDA

*** 최종데이터셋** → 총 234,573개의 데이터

NEWTIME	전류평균	온도	선간전압평균	상전압평균	유효전력평균	무효전력평균	누적전력량	주파수	전압고조파평균	역률평균
2020-10-22 00:00:22	70.500000	28.125	387.250000	223.500000	44696.0	14954.0	2.702882e+05	59.872740	4.817708	0.948331
2020-10-22 00:01:22	31.822916	28.125	387.416656	223.500000	19313.0	9604.5	2.708402e+05	59.863403	4.817708	0.895390
2020-10-22 00:02:22	70.927086	28.125	387.083344	223.333328	44907.0	15036.0	2.714893e+05	59.910122	4.785156	0.948258
2020-10-22 00:03:22	72.843750	27.500	387.500000	223.583328	46233.0	15494.0	2.722598e+05	59.872740	4.817708	0.948171
2020-10-22 00:04:22	47.104168	28.750	387.666656	223.750000	30076.0	11194.0	2.729348e+05	59.891440	4.817708	0.937192

ex 공기압축기

*** 표준화**

- StandardScaler() 사용

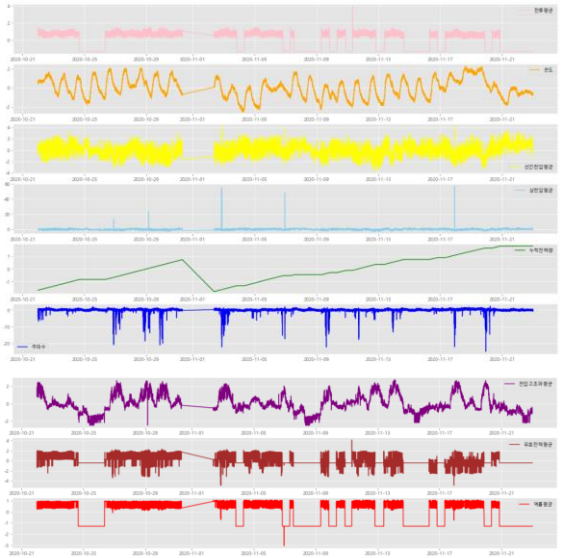
NEWTIME	전류평균	온도	선간전압평균	상전압평균	유효전력평균	무효전력평균	누적전력량	주파수	전압고조파평균	역률평균
2020-10-22 00:00:22	0.750510	0.844303	0.099383	0.113618	0.769216	0.720338	-1.457528	0.246067	1.581024	0.513532
2020-10-22 00:01:22	-0.636965	0.844303	0.134519	0.113618	-0.676130	-0.175280	-1.457409	-0.067683	1.581024	0.373486
2020-10-22 00:02:22	0.765831	0.844303	0.064246	0.052298	0.781231	0.734066	-1.457268	1.502209	1.548855	0.513338
2020-10-22 00:03:22	0.834588	0.688688	0.152091	0.144275	0.856735	0.810745	-1.457101	0.246067	1.581024	0.513108
2020-10-22 00:04:22	-0.088776	0.999917	0.187228	0.205595	-0.063268	0.090835	-1.456955	0.874440	1.581024	0.484065

02 전처리 및 EDA

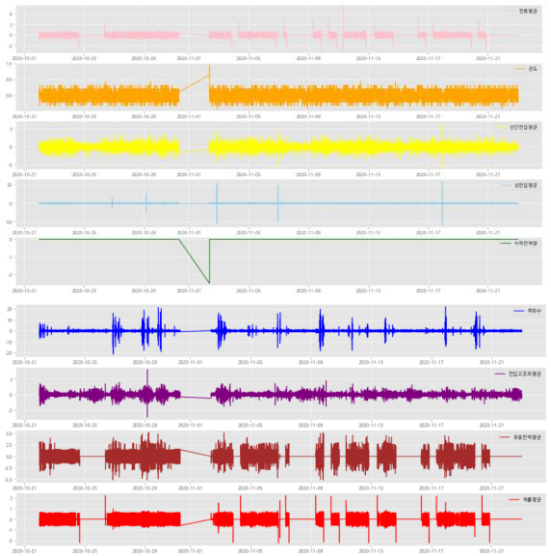
* 설비 기계별 변수들의 분포 시각화

공조설비

집진기2 표준화 후

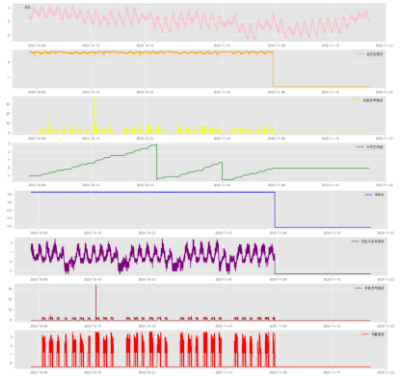


집진기2 정상화 후

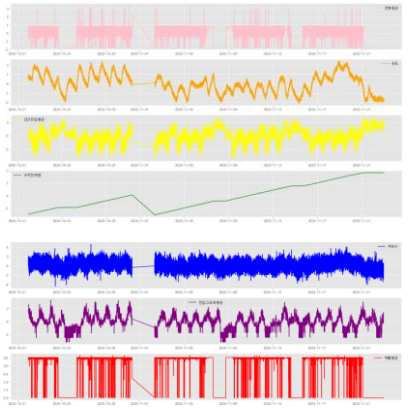


변수들 간의 분포는 각 설비별로 비슷한 형태임을 파악

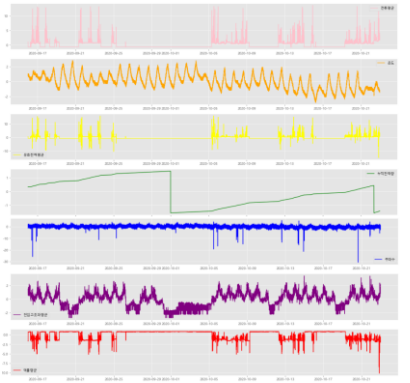
펌프/일반모터



공기압축기



공작기계



03 분석내용 및 결과해석

VIF - 다중공선성 제거

펌프/일반모터
고주파발생기_1

	온도	상전압평균	유효전력평균	누적전력량	주파수	전압고조파평균	무효전력평균	역률평균
온도	1.00	-0.34	0.18	0.11	0.50	-0.24	0.25	0.22
상전압평균	-0.34	1.00	-0.58	-0.01	-0.18	0.55	-0.66	-0.66
유효전력평균	0.18	-0.58	1.00	0.08	0.10	-0.32	0.84	0.95
누적전력량	0.11	-0.01	0.08	1.00	0.04	0.06	0.06	0.07
주파수	0.50	-0.18	0.10	0.04	1.00	-0.15	0.13	0.12
전압고조파평균	-0.24	0.55	-0.32	0.06	-0.15	1.00	-0.38	-0.37
무효전력평균	0.25	-0.66	0.84	0.06	0.13	-0.38	1.00	0.93
역률평균	0.22	-0.66	0.95	0.07	0.12	-0.37	0.93	1.00

최종 사용 독립변수

	VIF_Factor	Feature
3	1.028648	누적전력량
4	1.337865	주파수
5	1.460767	전압고조파평균
0	1.486828	온도
1	2.265512	상전압평균
2	3.457441	유효전력평균
6	4.059122	무효전력평균

공조설비
집진기2

	전류평균	온도	선간전압평균	상전압평균	누적전력량	주파수	전압고조파평균	무효전력평균	역률평균
전류평균	1.00	0.34	-0.21	-0.13	-0.24	0.02	0.51	0.35	0.96
온도	0.34	1.00	-0.14	-0.10	0.12	0.21	0.02	0.00	0.39
선간전압평균	-0.21	-0.14	1.00	0.79	0.08	0.10	-0.34	-0.03	-0.19
상전압평균	-0.13	-0.10	0.79	1.00	0.05	0.05	-0.23	-0.07	-0.13
누적전력량	-0.24	0.12	0.08	0.05	1.00	0.02	-0.02	-0.31	-0.19
주파수	0.02	0.21	0.10	0.05	0.02	1.00	-0.09	0.16	0.06
전압고조파평균	0.51	0.02	-0.34	-0.23	-0.02	-0.09	1.00	0.24	0.47
무효전력평균	0.35	0.00	-0.03	-0.07	-0.31	0.16	0.24	1.00	0.23
역률평균	0.96	0.39	-0.19	-0.13	-0.19	0.06	0.47	0.23	1.00

	VIF_Factor	Feature
5	1.107512	주파수
3	1.237369	누적전력량
7	1.255046	무효전력평균
0	1.382476	온도
6	1.625656	전압고조파평균
4	1.738901	역률평균
2	2.754299	상전압평균
1	3.058200	선간전압평균

03 분석내용 및 결과해석

VIF - 다중공선성 제거

공기압축기 컴프레서_1

	전류평균	온도	선간전압평균	누적전력량	주파수	전압고조파평균
전류평균	1.00	0.44	-0.55	-0.19	0.22	0.14
온도	0.44	1.00	-0.46	-0.09	0.46	0.04
선간전압평균	-0.55	-0.46	1.00	0.22	-0.20	0.23
누적전력량	-0.19	-0.09	0.22	1.00	-0.03	0.01
주파수	0.22	0.46	-0.20	-0.03	1.00	-0.02
전압고조파평균	0.14	0.04	0.23	0.01	-0.02	1.00

최종 사용 독립변수

	VIF_Factor	Feature
3	1.058778	누적전력량
5	1.189768	전압고조파평균
4	1.278273	주파수
1	1.645869	온도
0	1.686613	전류평균
2	1.868338	선간전압평균

공작기계 KBV 2호기

	전류평균	온도	유효전력평균	누적전력량	주파수	전압고조파평균
전류평균	1.00	-0.19	0.90	-0.11	-0.13	0.44
온도	-0.19	1.00	-0.16	0.12	0.55	-0.41
유효전력평균	0.90	-0.16	1.00	-0.09	-0.10	0.40
누적전력량	-0.11	0.12	-0.09	1.00	0.04	0.07
주파수	-0.13	0.55	-0.10	0.04	1.00	-0.25
전압고조파평균	0.44	-0.41	0.40	0.07	-0.25	1.00

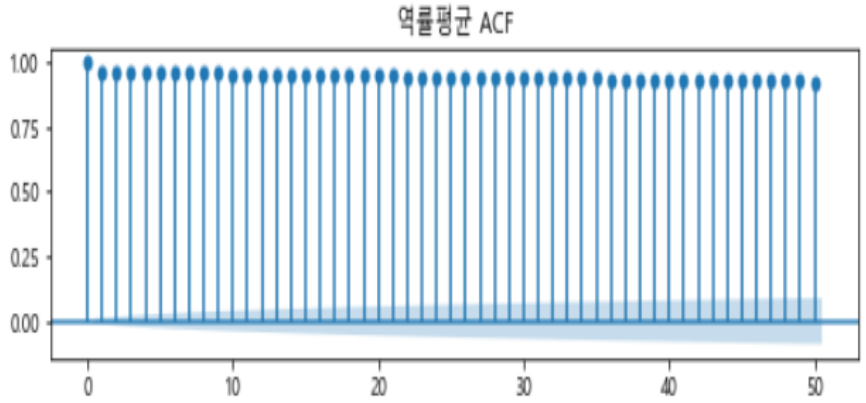
	VIF_Factor	Feature
3	1.058654	누적전력량
4	1.432229	주파수
5	1.490531	전압고조파평균
1	1.643785	온도
2	5.396219	유효전력평균
0	5.709995	전류평균

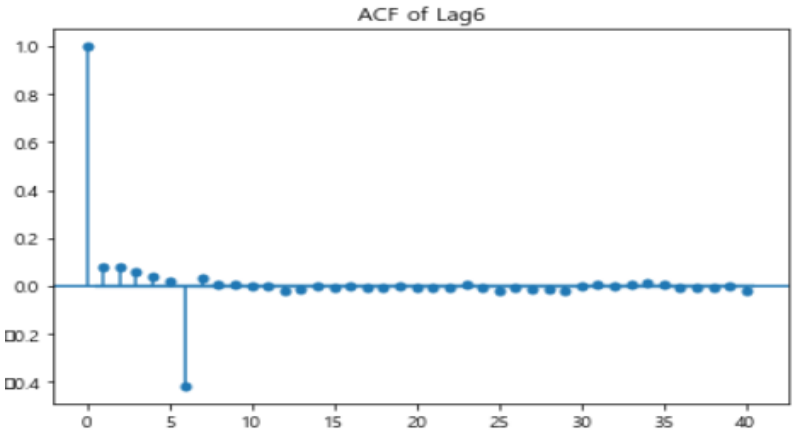
03 분석내용 및 결과해석

ADF 검정, KPSS 검정

ex 공기압축기 역률평균 p-value

p-value	raw-data	추세제거	계절성제거(lag6)	계절성제거 (lag12)
ADF 검정 (유의수준보다 작으면 정상)	0.006878	0.004278	0	0
KPSS (유의수준보다 크면 정상)	0.01	0.01	0.1	0.1

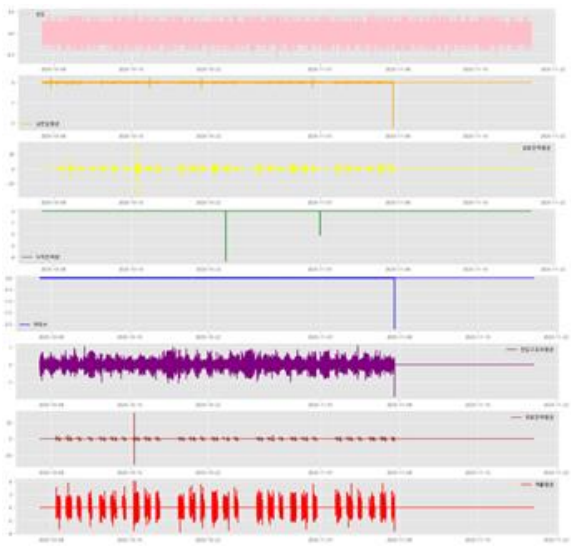




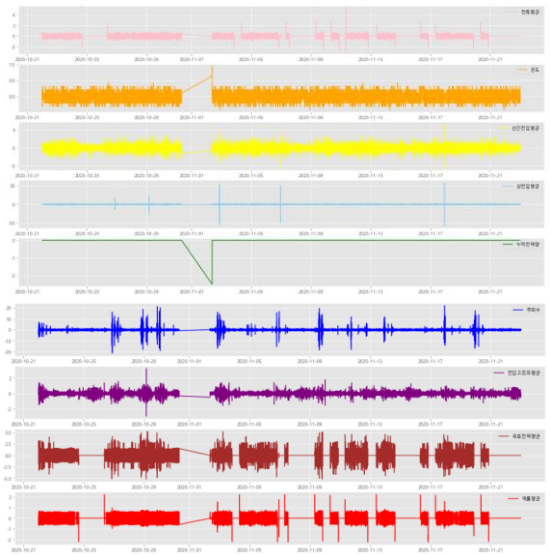
03 분석내용 및 결과해석

* 정상화 후 변수 분포 시각화

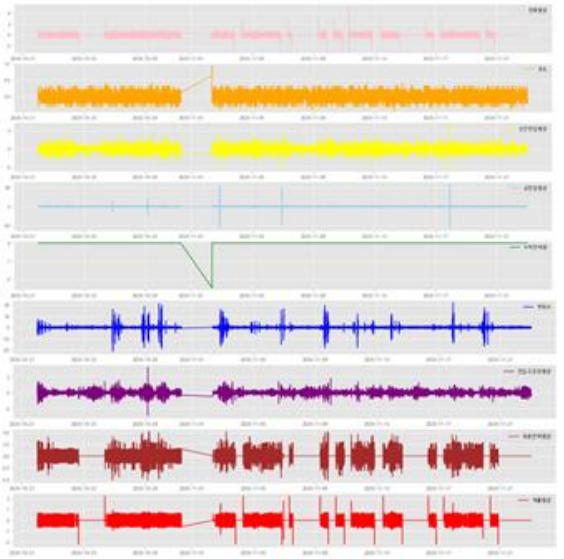
펌프/일반모터



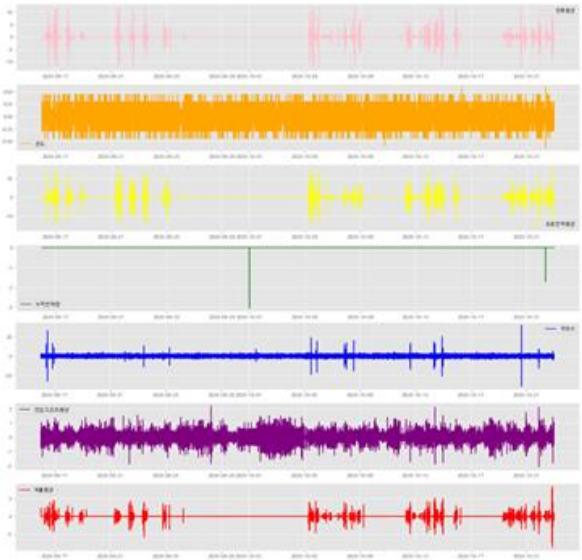
공조설비



공기압축기



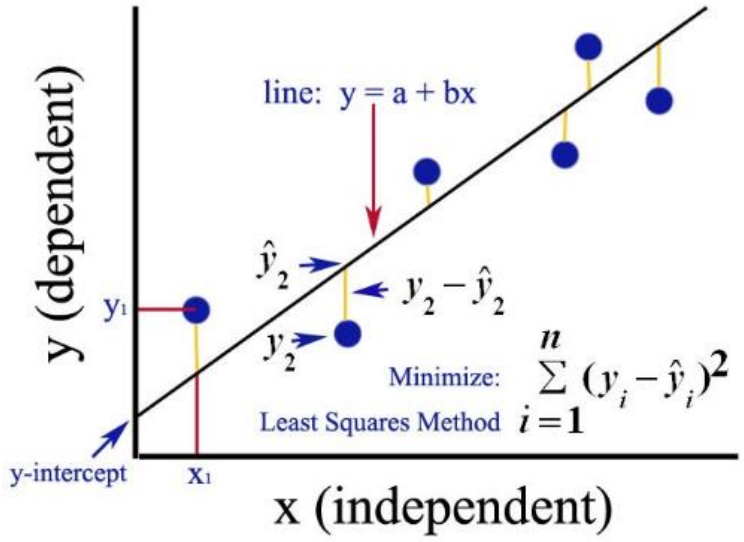
공작기계



03 분석내용 및 결과해석

OLS

* **OLS** (Ordinary Least Squares)



- 최소자승법(OLS)은 잔차제곱합(RSS: Residual Sum of Squares)을 최소화하는 가중치 벡터를 구하는 방법
- 모델에 대한 해석이 쉬움

03 분석내용 및 결과해석

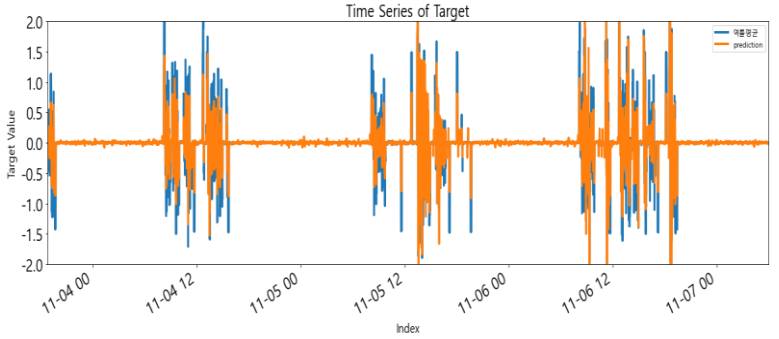
OLS

* 펌프/일반모터 - 고주파전압기1

OLS Regression Results

Dep. Variable:	역률평균	**R-squared:**	0.734
Model:	OLS	**Adj. R-squared:**	0.734
Method:	Least Squares	**F-statistic:**	1.580e+04
Date:	Mon, 13 Sep 2021	**Prob (F-statistic):**	0.00
Time:	20:31:18	**Log-Likelihood:**	3181.8
No. Observations:	39994	**AIC:**	-6348.
Df Residuals:	39986	**BIC:**	-6279.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	9.146e-05	0.001	0.082	0.935	-0.002	0.002
온도	-0.0058	0.007	-0.859	0.390	-0.019	0.007
상전압평균	-0.0693	0.004	-19.372	0.000	-0.076	-0.062
유효전력평균	0.5398	0.002	285.993	0.000	0.536	0.543
누적전력량	-0.0698	0.026	-2.690	0.007	-0.121	-0.019
주파수	0.0004	0.001	0.463	0.644	-0.001	0.002
전압고조파평균	-0.0171	0.004	-4.619	0.000	-0.024	-0.010
무효전력평균	0.0843	0.003	24.943	0.000	0.078	0.091



	MAE	MSE
Train	0.0753	0.0499
Test	0.0644	0.0264

유효전력평균이 역률 평균에 가장 많은 영향을 미침
-> 유효전력평균을 높임으로써 역률 평균값 증가시킬 수 있음

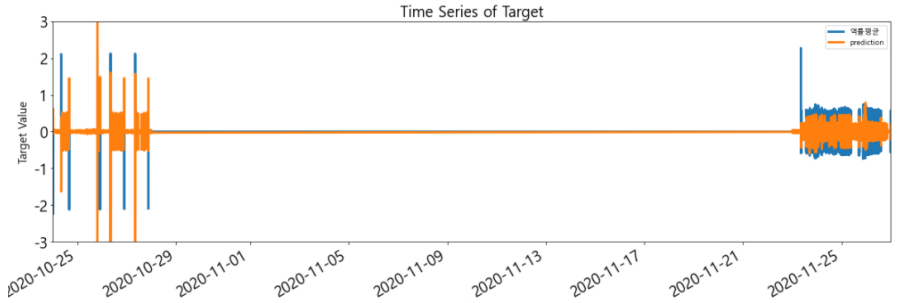
상전압평균, 누적 전력량, 전압고조파 평균은 역률 평균과 음의 상관관계를 가지고 있음
-> 전압고조파 평균을 낮춤으로써 역률 평균값을 증가시킬 수 있음

03 분석내용 및 결과해석

OLS

* 공조설비 - 집진기2

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0002	0.001	-0.201	0.841	-0.003	0.002
온도	-0.0158	0.012	-1.371	0.171	-0.038	0.007
선간전압평균	0.0291	0.002	12.515	0.000	0.025	0.034
상전압평균	-0.0325	0.002	-21.474	0.000	-0.036	-0.030
누적전력량	-0.1096	0.042	-2.638	0.008	-0.191	-0.028
주파수	0.0202	0.001	18.938	0.000	0.018	0.022
전압고조파평균	0.0617	0.005	11.459	0.000	0.051	0.072
무효전력평균	-0.1519	0.001	-107.302	0.000	-0.155	-0.149



	MAE	MSE
Train	0.1243	0.0647
Test	0.0817	0.0843

전압고조파 평균이 역률 평균에 가장 많은 영향을 미침

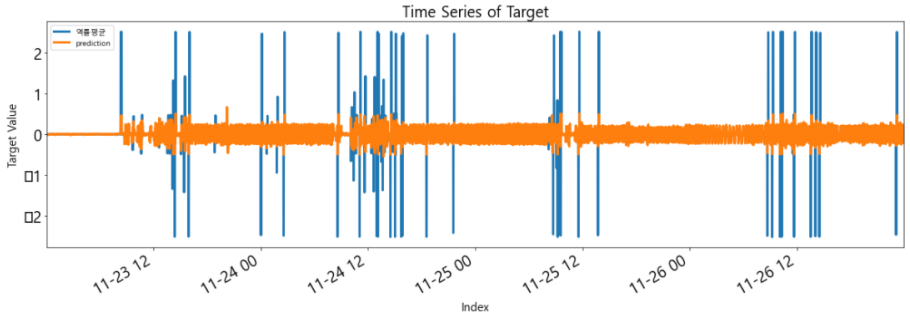
상전압평균, 누적 전력량, 무효전력평균 이 역률 평균과 음의 상관관계를 가지고 있음
-> 무효전력평균을 낮춤으로써 역률 평균값을 증가시킬 수 있음

03 분석내용 및 결과해석

OLS

* 공기압축기 - 컴프레서1

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0008	0.001	-0.667	0.505	-0.003	0.002
전압고조파평균	-0.0134	0.003	-4.330	0.000	-0.019	-0.007
전류평균	0.1894	0.002	101.516	0.000	0.186	0.193
온도	0.0071	0.008	0.915	0.360	-0.008	0.022
선간전압평균	-0.0027	0.003	-0.921	0.357	-0.008	0.003
누적전력량	1.1439	0.065	17.495	0.000	1.016	1.272
주파수	0.0008	0.001	0.766	0.444	-0.001	0.003



	MAE	MSE
Train	0.0808	0.0639
Test	0.0959	0.0628

누적 전력량, 전류평균이 역률 평균에 많은 영향을 미침
주파수는 역률 평균에 거의 영향을 미치지 않음

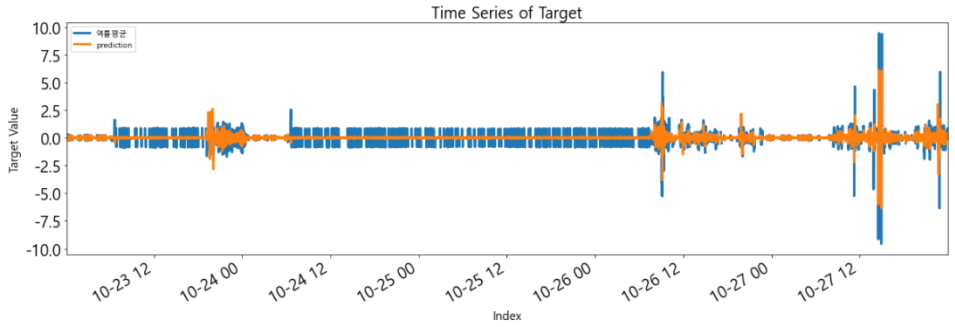
전압고조파 평균은 역률 평균과 음의 상관관계를 가지고 있음
-> 전압고조파 평균을 낮춤으로써 역률 평균값을 증가시킬 수 있음

03 분석내용 및 결과해석

OLS

* 공작기계 - KBV2호기

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.616e-05	0.001	-0.014	0.989	-0.002	0.002
전압고조파평균	0.0059	0.004	1.595	0.111	-0.001	0.013
전류평균	-0.0898	0.003	-35.116	0.000	-0.095	-0.085
온도	0.0018	0.008	0.235	0.814	-0.014	0.017
유효전력평균	0.3308	0.002	163.539	0.000	0.327	0.335
누적전력량	0.0523	0.032	1.649	0.099	-0.010	0.115
주파수	0.0147	0.001	14.449	0.000	0.013	0.017



	MAE	MSE
Train	0.0830	0.0712
Test	0.1324	0.1069

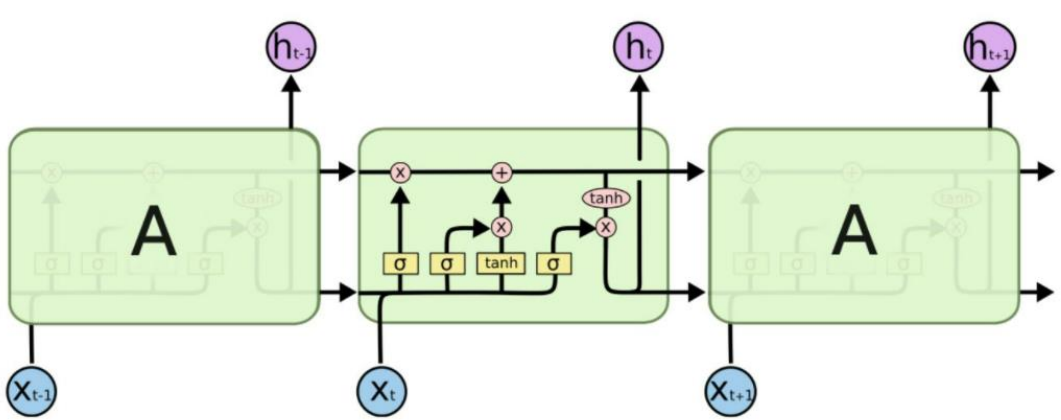
유효전력평균이 역률 평균에 가장 많은 영향을 미침
-> 유효전력평균을 높임으로써 역률 평균값 증가시킬 수 있음

전류평균은 역률 평균과 음의 상관관계를 가지고 있음

03 분석내용 및 결과해석

LSTM

* LSTM



- 기존의 RNN이 출력과 먼 위치에 있는 정보를 기억할 수 없다는 단점을 보완하여 장/단기 기억을 가능하게 설계한 신경망의 구조
- 순환 신경망(RNN; Recurrent Neural Networks)은 인공 신경망의 한 종류로서, 내부의 순환 구조가 포함되어 있기 때문에 시간에 의존적이거나 순차적인 데이터(Sequential data) 학습에 활용
- 주로 시계열 처리나, 자연어 처리에 사용

03 분석내용 및 결과해석

LSTM

* Time 기준 설정

past_history = 720
future_target = 72
STEP = 6

→ 720분 간의 히스토리를 기준으로 6분 간격으로 해당 시점으로부터 72분동안을 예측

* LSTM 파라미터 학습

BUFFER_SIZE = 1000
BATCH_SIZE = 1000
EPOCHS = 50
EVALUATION_INTERVAL = 10

```
multi_step_history = multi_step_model.fit(train_data_multi, epochs=EPOCHS,
                                          steps_per_epoch=EVALUATION_INTERVAL,
                                          validation_data=val_data_multi,
                                          validation_steps=50)
```

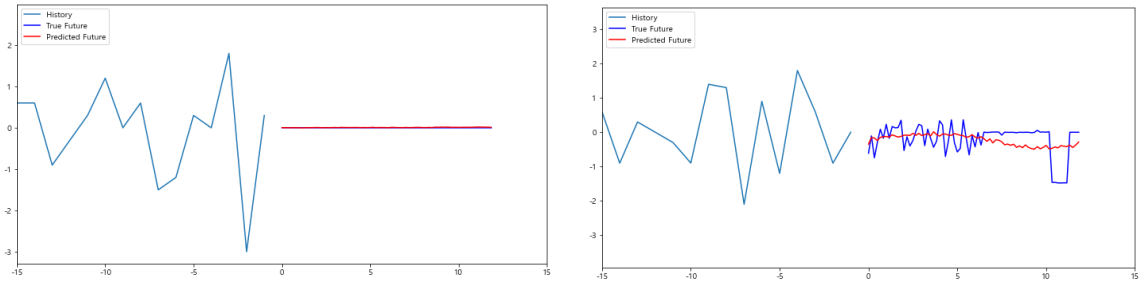
```
multi_step_model = tf.keras.models.Sequential()
multi_step_model.add(tf.keras.layers.LSTM(32, return_sequences=True,input_shape=x_train_multi.shape[-2:]))
multi_step_model.add(tf.keras.layers.LSTM(16, activation='relu'))
multi_step_model.add(tf.keras.layers.Dropout(0.3))
multi_step_model.add(tf.keras.layers.Dense(72))

multi_step_model.compile(optimizer='adam', loss='mean_squared_error',metrics=['mae'])
```

03 분석내용 및 결과해석

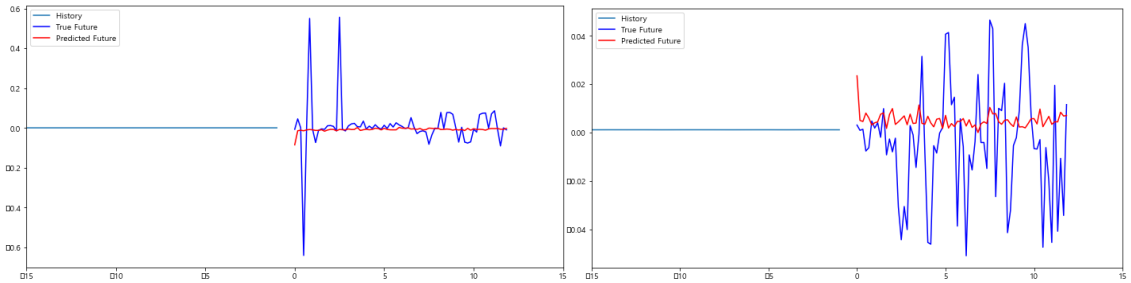
LSTM

펌프 **고주파전압기1**



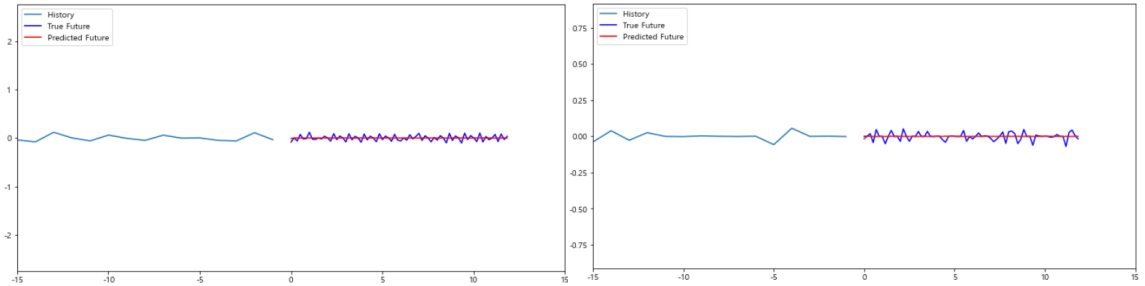
loss_MSE : 0.1635
MAE : 0.1173
y_val_multi : (-2.536445e-18, 3.2782)

공조설비 **집진기2**



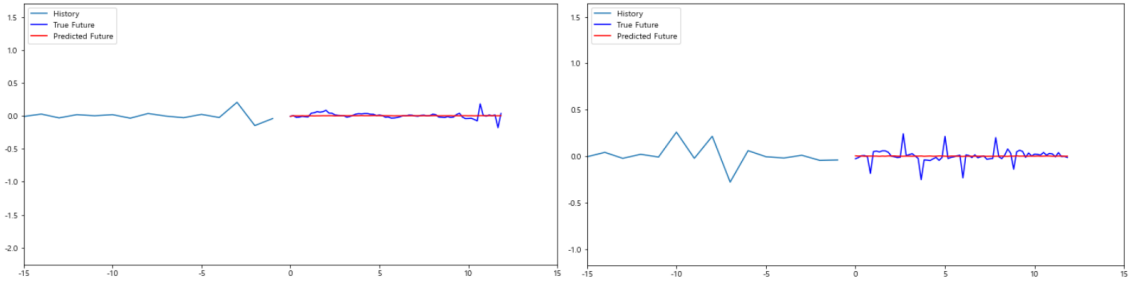
loss_MSE : 0.0465
MAE : 0.0804
y_val_multi : (-2.2359,2.1191)

공기압축기 **컴프레서1**



loss_MSE : 0.07813
MAE : 0.0674
y_val_multi : (-2.5204, 2.5175)

공작기계 **KBV 2호기**

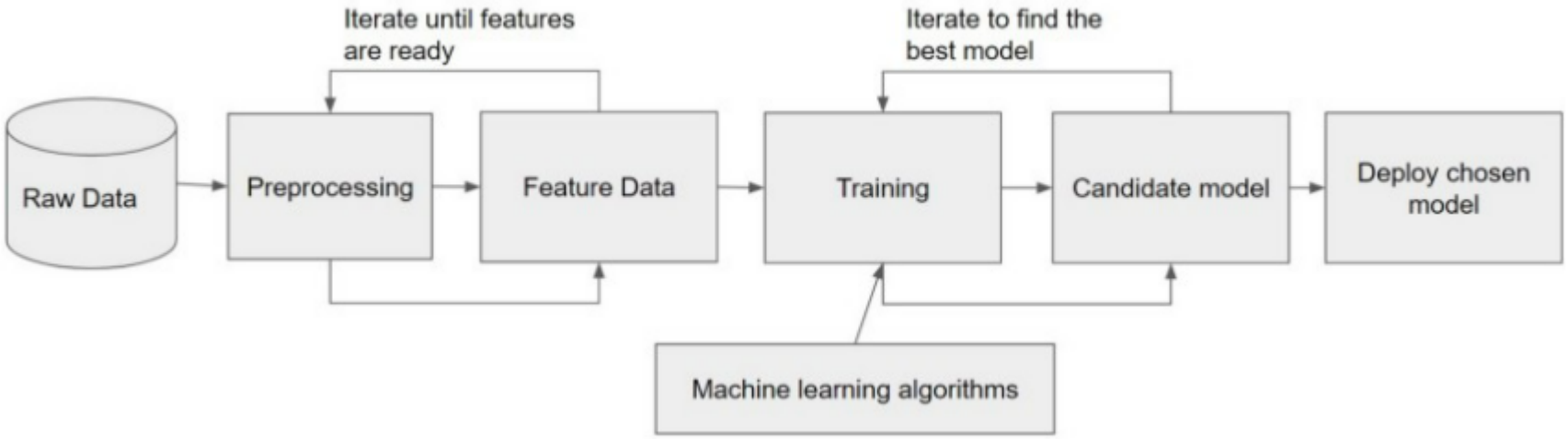


loss_MSE : 0.2297
MAE : 0.1753
y_val_multi : (-0.0019, 9.4318)

03 분석내용 및 결과해석

AutoML

* AutoML



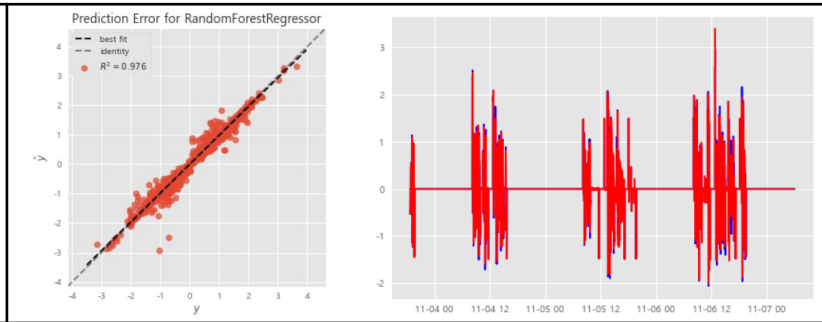
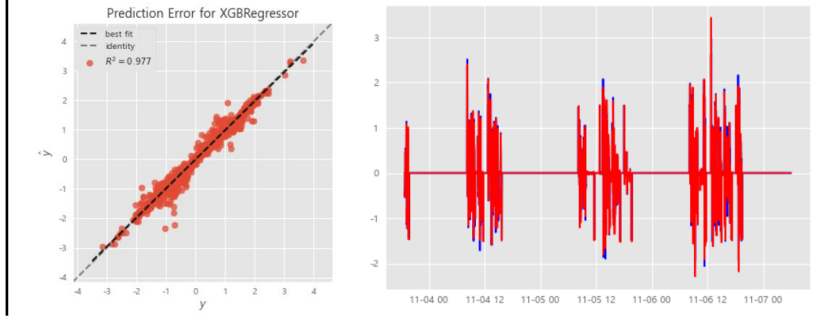
- 머신러닝과 딥러닝 모델을 구축하는 데 있어 기술력을 갖춘 데이터 과학자란 필요조건을 제거하고 레이블링된 학습 데이터를 입력으로 제공하고 최적화된 모델을 출력
- 소프트웨어에서 단순히 데이터에 대해 모든 종류의 모델을 학습시킨 다음 가장 결과가 좋은 모델을 선택하거나 다른 모델을 결합하는 하나 이상의 앙상블 모델을 만드는 방법으로 사용

03 분석내용 및 결과해석

AutoML

* 펌프/일반모터

고조파전압기1

Random Forest Regressor		
Extreme Gradient Boosting		

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	0.0170	0.0041	0.0635	0.9780	0.0339	0.2212	2.3620
rf	Random Forest Regressor	0.0168	0.0043	0.0646	0.9771	0.0340	0.2186	5.1490
xgboost	Extreme Gradient Boosting	0.0176	0.0044	0.0656	0.9764	0.0343	0.4133	1.3210
lightgbm	Light Gradient Boosting Machine	0.0178	0.0046	0.0670	0.9754	0.0344	0.3768	0.2170
catboost	CatBoost Regressor	0.0178	0.0047	0.0676	0.9749	0.0339	0.7236	8.1980
gbr	Gradient Boosting Regressor	0.0224	0.0051	0.0713	0.9724	0.0387	0.7599	1.4160
dt	Decision Tree Regressor	0.0215	0.0078	0.0877	0.9582	0.0461	0.3005	0.0900
huber	Huber Regressor	0.0317	0.0216	0.1133	0.4840	0.0483	0.9210	0.4370
knn	K Neighbors Regressor	0.0389	0.0131	0.1143	0.9293	0.0652	0.6220	0.6300
ada	AdaBoost Regressor	0.0547	0.0165	0.1284	0.9112	0.0722	1.1788	0.4870
br	Bayesian Ridge	0.0713	0.0531	0.2056	0.7197	0.0897	0.9617	0.0180
ridge	Ridge Regression	0.0713	0.0531	0.2056	0.7197	0.0897	0.9617	0.0150
lr	Linear Regression	0.0712	0.0531	0.2056	0.7197	0.0897	0.9618	0.9320
lar	Least Angle Regression	0.0712	0.0531	0.2056	0.7197	0.0897	0.9618	0.0160
omp	Orthogonal Matching Pursuit	0.0673	0.0475	0.2092	0.7469	0.0949	1.0068	0.0150
par	Passive Aggressive Regressor	0.2167	0.1629	0.3637	0.1410	0.1933	14.0079	0.0400
lasso	Lasso Regression	0.1643	0.1860	0.4311	-0.0003	0.2758	1.0256	0.0370
en	Elastic Net	0.1643	0.1860	0.4311	-0.0003	0.2758	1.0256	0.0170
llar	Lasso Least Angle Regression	0.1643	0.1860	0.4311	-0.0003	0.2758	1.0256	0.0170

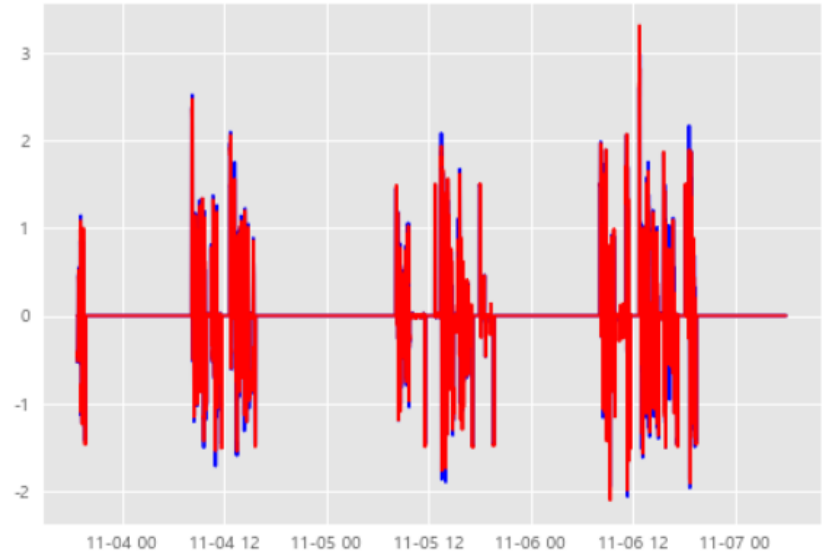
03 분석내용 및 결과해석

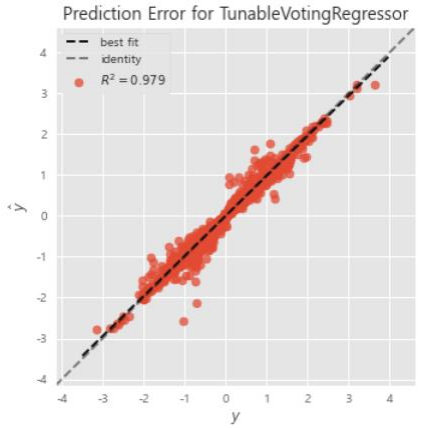
AutoML

*** 펌프/일반모터**

- RMSE 기준으로 상위 5개 모델 중 train의 R^2와 test의 R^2의 차이가 적은 모델 3개 앙상블

Blending(rf, xgb, lightgbm)	
RandomForest Regressor	RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse', max_depth=None, max_features='auto', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1, oob_score=False, random_state=6, verbose=0, warm_start=False)
XGBRegressor	XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1, importance_type='gain', interaction_constraints='', learning_rate=0.300000012, max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=-1, num_parallel_tree=1, objective='reg:squarederror', random_state=6, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='auto', validate_parameters=1, verbosity=0)
LGBMRegressor	LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0, importance_type='split', learning_rate=0.1, max_depth=-1, min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0, n_estimators=100, n_jobs=-1, num_leaves=31, objective=None, random_state=6, reg_alpha=0.0, reg_lambda=0.0, silent=True, subsample=1.0, subsample_for_bin=200000, subsample_freq=0)





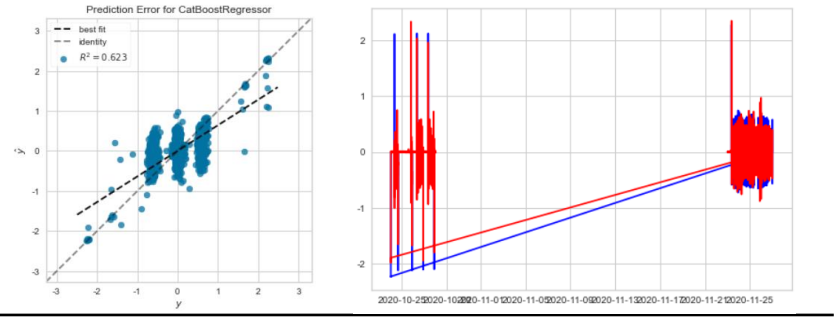
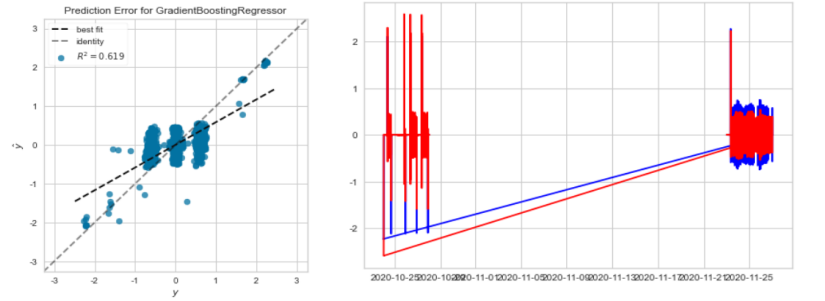
	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	0.0160	0.0036	0.0598	0.9815	0.0327	0.3881
1	0.0157	0.0029	0.0534	0.9860	0.0280	0.3725
2	0.0164	0.0036	0.0603	0.9811	0.0324	0.2437
3	0.0170	0.0036	0.0602	0.9799	0.0323	0.3232
4	0.0178	0.0038	0.0617	0.9790	0.0333	0.2503
5	0.0155	0.0036	0.0602	0.9793	0.0318	0.2997
6	0.0183	0.0080	0.0895	0.9588	0.0382	0.2948
7	0.0159	0.0035	0.0595	0.9809	0.0327	0.2551
8	0.0176	0.0043	0.0655	0.9758	0.0349	0.2681
9	0.0156	0.0035	0.0592	0.9802	0.0324	0.2924
Mean	0.0166	0.0040	0.0629	0.9782	0.0329	0.2988
SD	0.0010	0.0014	0.0093	0.0069	0.0024	0.0472

03 분석내용 및 결과해석

AutoML

* 공조설비

집진기2

CatBoost Regressor		
Gradient Boosting Regressor		

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	0.0812	0.0328	0.1808	0.6019	0.1110	7.7122	3.9760
catboost	CatBoost Regressor	0.0833	0.0332	0.1822	0.5959	0.1102	10.7480	8.1340
rf	Random Forest Regressor	0.0808	0.0338	0.1836	0.5892	0.1113	8.0421	11.3030
lightgbm	Light Gradient Boosting Machine	0.0827	0.0339	0.1839	0.5884	0.1120	8.9484	0.2250
gbr	Gradient Boosting Regressor	0.0866	0.0345	0.1857	0.5798	0.1159	7.1962	1.8860
xgboost	Extreme Gradient Boosting	0.0828	0.0346	0.1859	0.5787	0.1107	11.6683	1.4970
knn	K Neighbors Regressor	0.1040	0.0529	0.2298	0.3570	0.1282	7.7365	0.1970
lar	Least Angle Regression	0.1322	0.0627	0.2503	0.2386	0.1377	20.9289	0.0280
br	Bayesian Ridge	0.1322	0.0627	0.2503	0.2386	0.1377	20.8649	0.0320
lr	Linear Regression	0.1322	0.0627	0.2503	0.2386	0.1377	20.9289	0.9650
ridge	Ridge Regression	0.1322	0.0627	0.2503	0.2386	0.1377	20.9182	0.0290
omp	Orthogonal Matching Pursuit	0.1185	0.0657	0.2562	0.2023	0.1354	6.3728	0.0260
dt	Decision Tree Regressor	0.0929	0.0659	0.2564	0.1979	0.1378	11.2175	0.2050
huber	Huber Regressor	0.1139	0.0696	0.2636	0.1551	0.1171	9.1470	0.3040
ada	AdaBoost Regressor	0.1439	0.0717	0.2677	0.1275	0.2078	3.7754	1.3090
lasso	Lasso Regression	0.1337	0.0823	0.2868	-0.0002	0.2173	1.1055	0.0370
en	Elastic Net	0.1337	0.0823	0.2868	-0.0002	0.2173	1.1055	0.0280
llar	Lasso Least Angle Regression	0.1337	0.0823	0.2868	-0.0002	0.2173	1.1055	0.0260
par	Passive Aggressive Regressor	0.4047	0.5020	0.6626	-5.0615	0.3085	130.8389	0.0470

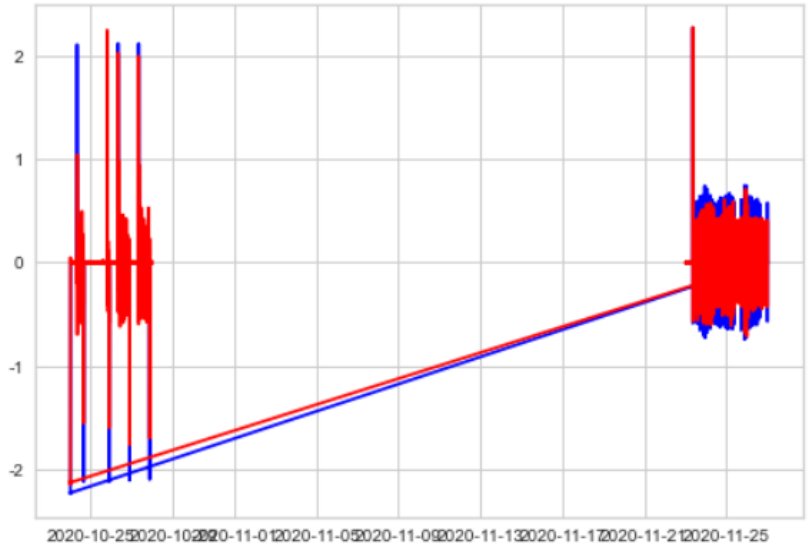
03 분석내용 및 결과해석

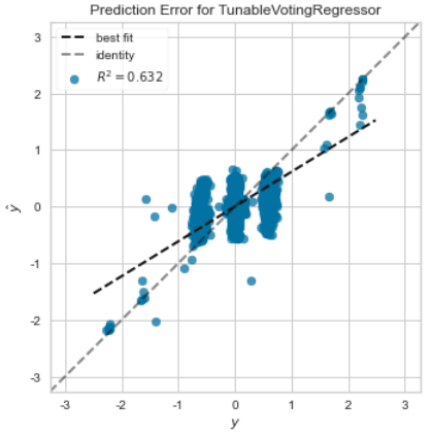
AutoML

*** 공조설비**

- RMSE 기준으로 상위 5개 모델 중 train의 R^2와 test의 R^2의 차이가 적은 모델 3개 앙상블

Blending(catboost, lightgbm, gbr)	
CatBoost Regressor	<catboost.core.CatBoostRegressor at 0x23e5e50a190>
LGBMRegressor	LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0, importance_type='split', learning_rate=0.1, max_depth=-1, min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0, n_estimators=100, n_jobs=-1, num_leaves=31, objective=None, random_state=6, reg_alpha=0.0, reg_lambda=0.0, silent=True, subsample=1.0, subsample_for_bin=200000, subsample_freq=0)
Gradient Boosting Regressor	GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse', init=None, learning_rate=0.1, loss='ls', max_depth=3, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_iter_no_change=None, presort='deprecated', random_state=6, subsample=1.0, tol=0.0001, validation_fraction=0.1, verbose=0, warm_start=False)





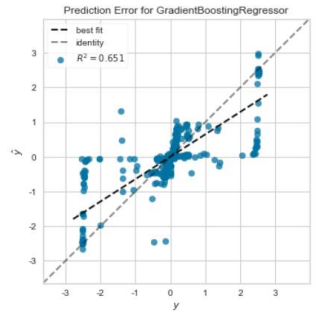
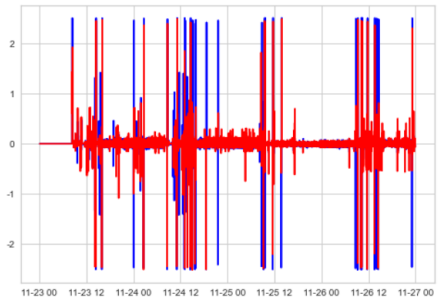
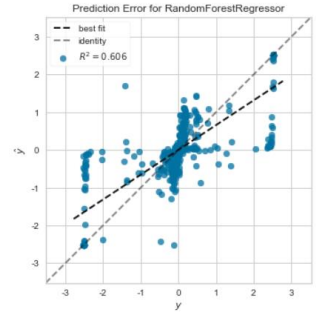
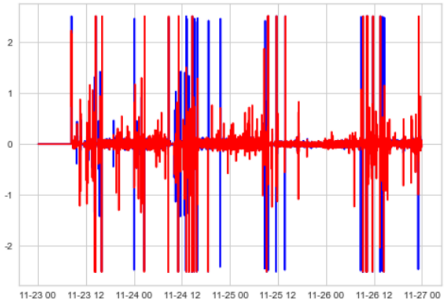
	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	0.0811	0.0305	0.1748	0.6123	0.1115	8.6022
1	0.0856	0.0359	0.1896	0.5695	0.1115	7.2268
2	0.0814	0.0328	0.1811	0.6198	0.1091	10.1952
3	0.0848	0.0340	0.1844	0.5928	0.1157	6.8685
4	0.0868	0.0340	0.1844	0.6352	0.1111	10.8566
5	0.0837	0.0324	0.1799	0.6066	0.1131	6.6157
6	0.0847	0.0333	0.1824	0.5993	0.1123	6.8011
7	0.0769	0.0292	0.1709	0.6388	0.1035	6.9453
8	0.0816	0.0310	0.1762	0.5911	0.1100	11.6265
9	0.0822	0.0313	0.1770	0.5877	0.1125	5.8411
Mean	0.0829	0.0324	0.1801	0.6053	0.1110	8.1579
SD	0.0027	0.0019	0.0052	0.0206	0.0030	1.9300

03 분석내용 및 결과해석

AutoML

* 공기압축기

컴프레서1

Gradient Boosting Regressor		
Random Forest Regressor		

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
gbr	Gradient Boosting Regressor	0.0252	0.0249	0.1570	0.6908	0.0739	2.9663	1.9420
et	Extra Trees Regressor	0.0260	0.0256	0.1592	0.6803	0.0786	4.1174	3.2300
rf	Random Forest Regressor	0.0256	0.0262	0.1612	0.6717	0.0788	3.9079	7.5020
xgboost	Extreme Gradient Boosting	0.0276	0.0296	0.1716	0.6281	0.0834	4.6163	1.5270
knn	K Neighbors Regressor	0.0321	0.0393	0.1968	0.5164	0.0893	3.7914	0.1130
lightgbm	Light Gradient Boosting Machine	0.0355	0.0414	0.2024	0.4859	0.0866	5.2772	0.2100
catboost	CatBoost Regressor	0.0359	0.0414	0.2025	0.4855	0.0847	10.0013	7.4500
dt	Decision Tree Regressor	0.0281	0.0426	0.2061	0.4627	0.0966	6.0786	0.1530
ada	AdaBoost Regressor	0.0774	0.0538	0.2302	0.3254	0.1318	15.3610	0.2800
lr	Linear Regression	0.0814	0.0642	0.2526	0.2052	0.1266	23.6918	1.2160
lar	Least Angle Regression	0.0814	0.0642	0.2526	0.2052	0.1266	23.6905	0.0160
ridge	Ridge Regression	0.0815	0.0642	0.2526	0.2052	0.1266	23.6084	0.0160
br	Bayesian Ridge	0.0814	0.0642	0.2526	0.2052	0.1266	23.6660	0.0190
omp	Orthogonal Matching Pursuit	0.0808	0.0648	0.2537	0.1986	0.1272	20.8766	0.0180
huber	Huber Regressor	0.0427	0.0730	0.2692	0.0995	0.1327	5.2760	0.2110
lasso	Lasso Regression	0.0549	0.0811	0.2837	-0.0005	0.1490	1.9615	0.0170
en	Elastic Net	0.0549	0.0811	0.2837	-0.0005	0.1490	1.9615	0.0160
llar	Lasso Least Angle Regression	0.0549	0.0811	0.2837	-0.0005	0.1490	1.9615	0.0160
par	Passive Aggressive Regressor	0.2453	0.2051	0.4110	-1.6339	0.2338	429.9728	0.0390

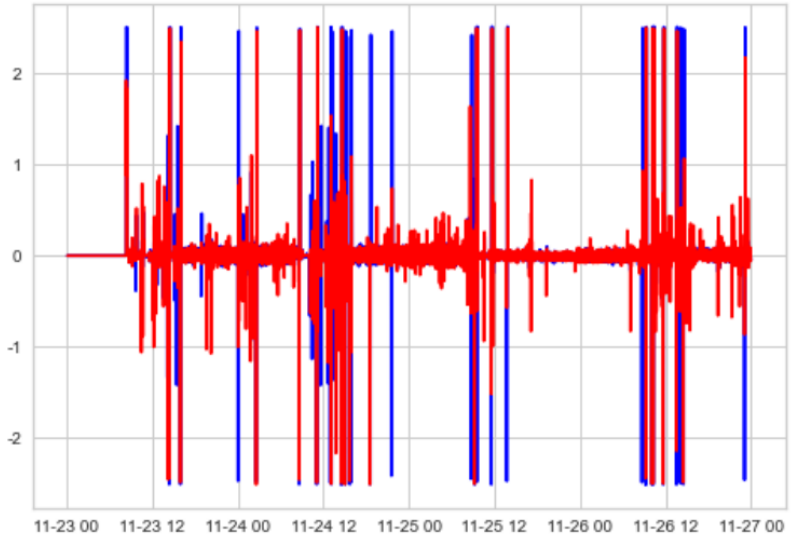
03 분석내용 및 결과해석

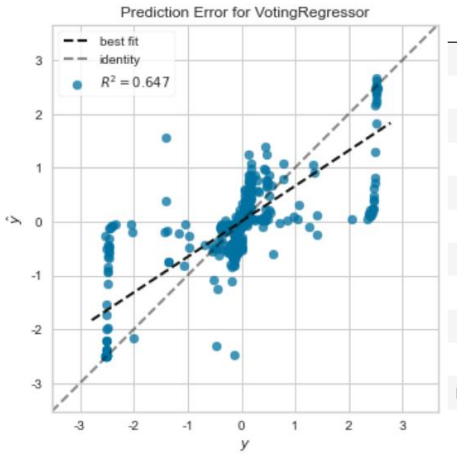
AutoML

* 공기압축기

- RMSE 기준으로 상위 5개 모델 중 train의 R^2와 test의 R^2의 차이가 적은 모델 3개 앙상블

Blending(gbr, et, rf)	
Gradient Boosting Regressor	GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse', init=None, learning_rate=0.1, loss='ls', max_depth=3, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_iter_no_change=None, presort='deprecated', random_state=6, subsample=1.0, tol=0.0001, validation_fraction=0.1, verbose=0, warm_start=False)
ExtraTreesRegressor	ExtraTreesRegressor(bootstrap=False, ccp_alpha=0.0, criterion='mse', max_depth=None, max_features='auto', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1, oob_score=False, random_state=6, verbose=0, warm_start=False)
RandomForest Regressor	RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse', max_depth=None, max_features='auto', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1, oob_score=False, random_state=6, verbose=0, warm_start=False)





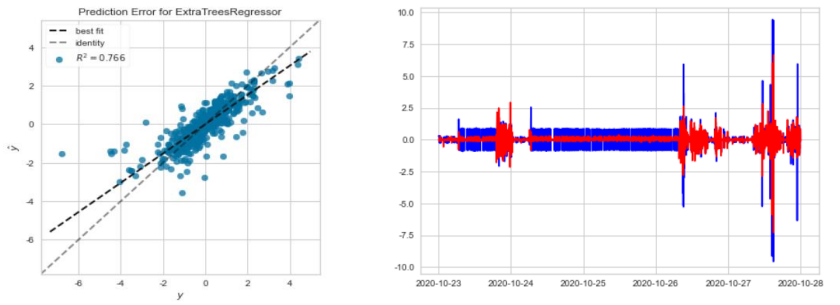
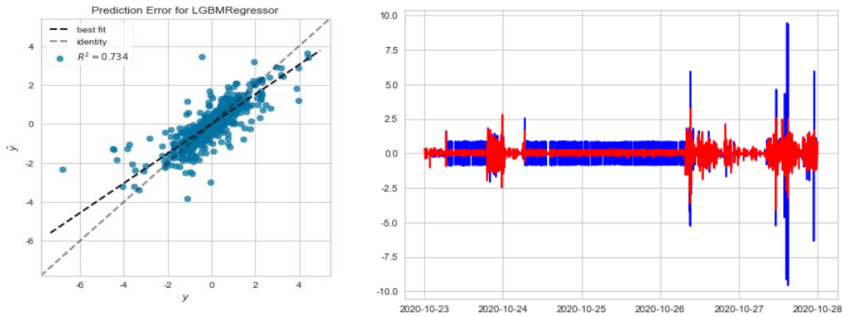
	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	0.0282	0.0328	0.1812	0.5735	0.0844	3.8204
1	0.0242	0.0247	0.1572	0.6144	0.0766	4.9132
2	0.0223	0.0175	0.1322	0.7442	0.0692	3.4751
3	0.0265	0.0292	0.1710	0.7153	0.0841	3.9111
4	0.0236	0.0226	0.1502	0.7492	0.0732	2.0529
5	0.0265	0.0280	0.1674	0.7050	0.0822	3.3939
6	0.0255	0.0248	0.1575	0.7229	0.0768	5.0412
7	0.0261	0.0289	0.1701	0.6776	0.0731	2.9463
8	0.0219	0.0188	0.1372	0.6664	0.0669	3.0882
9	0.0258	0.0274	0.1655	0.6496	0.0797	2.8180
Mean	0.0251	0.0255	0.1589	0.6818	0.0766	3.5460
SD	0.0019	0.0046	0.0146	0.0542	0.0057	0.8758

03 분석내용 및 결과해석

AutoML

* 공작기계

KBV2호기

Extra Trees Regressor		
Light Gradient Boosting Machine		

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
et	Extra Trees Regressor	0.0438	0.0290	0.1675	0.7615	0.0697	1.0475	5.1260
rf	Random Forest Regressor	0.0439	0.0304	0.1721	0.7483	0.0703	1.0161	8.3570
lightgbm	Light Gradient Boosting Machine	0.0483	0.0320	0.1767	0.7345	0.0735	0.9841	0.2370
catboost	CatBoost Regressor	0.0461	0.0323	0.1772	0.7323	0.0721	0.9775	8.5330
xgboost	Extreme Gradient Boosting	0.0456	0.0336	0.1802	0.7234	0.0737	0.9677	1.3190
gbr	Gradient Boosting Regressor	0.0545	0.0367	0.1887	0.6974	0.0832	1.0140	1.4510
knn	K Neighbors Regressor	0.0578	0.0403	0.1992	0.6624	0.0879	1.1127	0.1640
dt	Decision Tree Regressor	0.0571	0.0499	0.2213	0.5825	0.0859	1.4887	0.1500
ada	AdaBoost Regressor	0.0983	0.0710	0.2641	0.4052	0.1342	2.6631	0.9020
lr	Linear Regression	0.0834	0.0723	0.2674	0.3886	0.1228	1.3648	0.9540
lar	Least Angle Regression	0.0834	0.0723	0.2674	0.3886	0.1228	1.3648	0.0190
ridge	Ridge Regression	0.0834	0.0723	0.2674	0.3886	0.1228	1.3648	0.0160
br	Bayesian Ridge	0.0834	0.0723	0.2674	0.3886	0.1228	1.3647	0.0210
omp	Orthogonal Matching Pursuit	0.0790	0.0736	0.2701	0.3760	0.1216	0.9928	0.0160
huber	Huber Regressor	0.0758	0.0856	0.2900	0.2786	0.1094	1.0169	0.1780
lasso	Lasso Regression	0.1072	0.1178	0.3421	-0.0005	0.2048	1.0022	0.0180
en	Elastic Net	0.1072	0.1178	0.3421	-0.0005	0.2048	1.0022	0.0170
llar	Lasso Least Angle Regression	0.1072	0.1178	0.3421	-0.0005	0.2048	1.0022	0.0190
par	Passive Aggressive Regressor	0.3717	0.4909	0.6594	-3.1992	0.2898	22.3383	0.0410

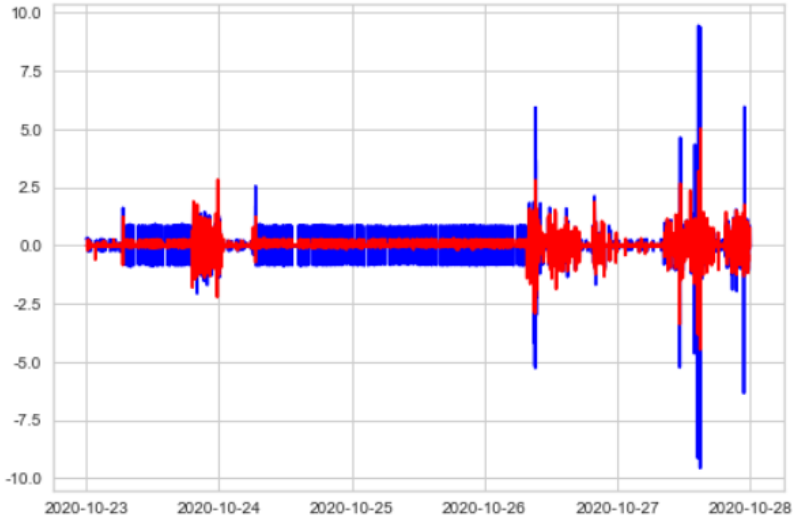
03 분석내용 및 결과해석

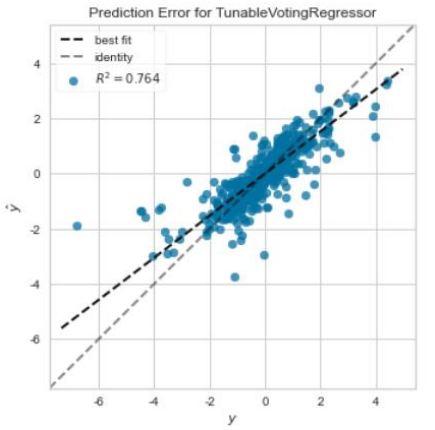
AutoML

*** 공작기계**

- RMSE 기준으로 상위 5개 모델 중 train의 R^2와 test의 R^2의 차이가 적은 모델 3개 앙상블

Blending(et, rf, lightgbm)	
ExtraTreesRegressor	ExtraTreeRegressor(bootstrap=False, ccp_alpha=0.0, criterion='mse', max_depth=None, max_features='auto', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1, oob_score=False, random_state=6, verbose=0, warm_start=False)
RandomForest Regressor	RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse', max_depth=None, max_features='auto', max_leaf_nodes=None, max_samples=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1, oob_score=False, random_state=6, verbose=0, warm_start=False)
LGBMRegressor	LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0, importance_type='split', learning_rate=0.1, max_depth=-1, min_child_samples=20, min_child_weight=0.001, min_split_gain=0.0, n_estimators=100, n_jobs=-1, num_leaves=31, objective=None, random_state=6, reg_alpha=0.0, reg_lambda=0.0, silent=True, subsample=1.0, subsample_for_bin=200000, subsample_freq=0)





	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	0.0479	0.0496	0.2226	0.6744	0.0734	0.9101
1	0.0423	0.0280	0.1672	0.7612	0.0671	0.9684
2	0.0446	0.0368	0.1919	0.7394	0.0734	0.9541
3	0.0475	0.0317	0.1781	0.7485	0.0764	1.0442
4	0.0419	0.0214	0.1464	0.7963	0.0680	0.9740
5	0.0463	0.0381	0.1953	0.7220	0.0724	0.9234
6	0.0436	0.0274	0.1655	0.7554	0.0685	0.9954
7	0.0430	0.0199	0.1411	0.8008	0.0635	0.9941
8	0.0395	0.0158	0.1257	0.8290	0.0627	0.9782
9	0.0393	0.0192	0.1384	0.7966	0.0645	0.9423
Mean	0.0436	0.0288	0.1672	0.7624	0.0690	0.9684
SD	0.0029	0.0099	0.0287	0.0429	0.0045	0.0369

03 분석내용 및 결과해석

* 결과해석

OLS				
	펌프/일반모터	공조설비	공기압축기	공작기계
MAE	0.06445	0.0985	0.0959	0.1324
MSE	0.0264	0.7719	0.0628	0.1069

LSTM				
	펌프/일반모터	공조설비	공기압축기	공작기계
MAE	0.1635	0.0465	0.0781	0.2297
MSE	0.1173	0.0804	0.0674	0.1753

Auto ML				
	펌프/일반모터	공조설비	공기압축기	공작기계
MAE	0.0166	0.0829	0.0252	0.0436
MSE	0.0040	0.0324	0.0248	0.0288

04 결론

- OLS, LSTM, AutoML 의 모델을 이용해 역률 평균 예측을 진행해본 결과,
  전반적으로 MSE는 AutoML을 이용한 앙상블 모델이 가장 낮았고
  공조설비의 경우 MAE는 LSTM이 더 낮아 성능이 좋다고 판단함.

- OLS를 통한 원인 분석을 진행한 결과, 전반적으로는 유효전력 평균과 전압고조파 평균이 역률 평균에 많은 영향을 미치는 것을 알 수 있었지만 설비별로 역률 평균에 영향을 미치는 요소가 다른 것을 파악함.

- 역률 평균에 영향을 미치는 특정 변수의 상관계수 부호가 설비별로 다른 경우가 존재하므로
  설비 각각의 특성에 맞는 역률 평균을 예측하고 영향을 미치는 요소들을 파악하는 것이 의미있다고 판단함.

- 설비별 실시간 역률값 예측과 원인 분석을 통해 역률을 개선함으로써 전력 손실과 전기 요금을 경감할 수 있고,
  설비 용량의 여유를 증가시킬 수 있으며 갑작스러운 가동중단으로 인한 막대한 손해를 방지할 수 있음.

05 활용방안

- 설비별 역률의 실시간 예측을 통한 설비 이상 알림 서비스와 이상 원인 진단 서비스 구현 가능

Q & A

06 사용한 코드

* json파일 읽어와 csv파일로 변환

```
## zip파일 읽기

import zipfile
import os
import pandas as pd
import json

my_zip = zipfile.ZipFile(r'C:₩Users₩aa₩Downloads₩전력 설비 에너지 패턴 및 고장 분석 센서₩Training₩[라벨링]3.공기압축기.zip')
my_zip.namelist()
```

```
list1=[]
list2=[]
list3=[]
final_list=[]

for title in my_zip.namelist():
    if ('3.공기압축기/2.SOH진단/1.역률평균/' in title) & (len(title)>=60):
        list1.append(title)
    elif ('3.공기압축기/2.SOH진단/2.전류고조파평균/' in title) & (len(title)>=60):
        list2.append(title)
    elif ('3.공기압축기/2.SOH진단/3.전압고조파평균/' in title) & (len(title)>=60):
        list3.append(title)

if (len(list1)==len(list2)) & (len(list2)==len(list3)):
    list1=sorted(list1)
    list2=sorted(list2)
    list3=sorted(list3)

    for pair in zip(list1,list2,list3):
        final_list.append(pair)
```

```
from tqdm import tqdm

for i in tqdm(range(len(final_list))):
    contents1=my_zip.read(final_list[i][0])
    contents2=my_zip.read(final_list[i][1])
    contents3=my_zip.read(final_list[i][2])

    json_data1=json.loads(contents1)
    json_data2=json.loads(contents2)
    json_data3=json.loads(contents3)

    data1=pd.DataFrame(json_data1)
    data2=pd.DataFrame(json_data2)
    data3=pd.DataFrame(json_data3)

    data1['ITEM_NAME']=data1['data'].apply(lambda x: x['ITEM_NAME'])
    data1['ITEM_VALUE']=data1['data'].apply(lambda x: x['ITEM_VALUE'])
    data1['TIMESTAMP']=data1['data'].apply(lambda x: x['TIMESTAMP'])
    data1['LABEL_NAME']=data1['data'].apply(lambda x: x['LABEL_NAME'])

    data2['ITEM_NAME']=data2['data'].apply(lambda x: x['ITEM_NAME'])
    data2['ITEM_VALUE']=data2['data'].apply(lambda x: x['ITEM_VALUE'])
    data2['TIMESTAMP']=data2['data'].apply(lambda x: x['TIMESTAMP'])
    data2['LABEL_NAME']=data2['data'].apply(lambda x: x['LABEL_NAME'])

    data3['ITEM_NAME']=data3['data'].apply(lambda x: x['ITEM_NAME'])
    data3['ITEM_VALUE']=data3['data'].apply(lambda x: x['ITEM_VALUE'])
    data3['TIMESTAMP']=data3['data'].apply(lambda x: x['TIMESTAMP'])
    data3['LABEL_NAME']=data3['data'].apply(lambda x: x['LABEL_NAME'])

    data1.drop('data',axis=1,inplace=True)
    data2.drop('data',axis=1,inplace=True)
    data3.drop('data',axis=1,inplace=True)

    data1=data1.rename(columns={'LABEL_NAME':'역률평균_LABEL'})
    data2=data2.rename(columns={'LABEL_NAME':'전류고조파평균_LABEL'})
    data3=data3.rename(columns={'LABEL_NAME':'전압고조파평균_LABEL'})

    data2=data2['전류고조파평균_LABEL']
    data3=data3[['전압고조파평균_LABEL']]

    data=pd.concat([data1,data2,data3],axis=1)
    data.drop('BASE_ITEM',axis=1,inplace=True)
    data.to_csv('공기압축기 {}.csv'.format(i))
```

```
final_data.to_csv('공기압축기_최종.csv')
```

06 사용한 코드

* 전처리

```
import os
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import numpy as np
import tensorflow as tf
import seaborn as sns
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go

import matplotlib.font_manager as fm

font_fname = 'C:/Windows/Fonts/Malgun.ttf' #적용할 폰트
font_family = fm.FontProperties(fname=font_fname).get_name() #폰트 설정
plt.rcParams["font.family"] = font_family  #폰트 적용

# 부호 오류 고치기 코드
import matplotlib
matplotlib.rcParams['axes.unicode_minus'] = False

import warnings
warnings.filterwarnings(action='ignore')

os.chdir(r"C:₩Users₩82109₩project₩2021 빅데이터 경진대회₩최종")
```

```
comp_1_one=pd.read_csv('comp1_concattt.csv')

TRAIN_SPLIT = 43135

comp_1_one.drop('Unnamed: 0', axis=1, inplace=True)

comp_1_one['TIMESTAMP'] = pd.to_datetime(comp_1_one['TIME'])
comp_1_one.drop('TIME', axis = 1, inplace=True)
comp_1_one.set_index('TIMESTAMP', inplace=True)

X = comp_1_one
X_train = comp_1_one[:43135]
X_test = comp_1_one[43135:]
X_train = X_train[['전류평균', '온도', '선간전압평균', '누적전력량', '주파수', '전압고조파평균','역률평균']]
X_test = X_test[['전류평균', '온도', '선간전압평균', '누적전력량', '주파수', '전압고조파평균','역률평균']]
```

06 사용한 코드

* 표준화

```
from sklearn import preprocessing

# 1. 스케일러를 선택 후 스케일러 객체를 지정한다
scaler = preprocessing.StandardScaler()

# 2. 스케일러 객체의 fit()함수를 이용하여 필요한 파라미터를 저장한다
# scaler_fit에는 칼럼별 최대와 최소값이 저장되어있다.
scaler_fit = scaler.fit(X)

# 3. 스케일러 객체의 transform()함수를 이용하여 스케일링을 수행한다
result=scaler_fit.transform(X)
```

```
def feature_engineering_scaling(scaler, X_train, X_test):
    # scaler파라미터는 아래 4개중 하나를 넣는다
    # preprocessing.MinMaxScaler()
    # preprocessing.StandardScaler()
    # preprocessing.RobustScaler()
    # preprocessing.Normalizer()
    scaler = scaler
    scaler_fit = scaler.fit(X_train)
    X_train_scaling = pd.DataFrame(scaler_fit.transform(X_train),
                                   index=X_train.index, columns=X_train.columns)
    X_test_scaling = pd.DataFrame(scaler_fit.transform(X_test),
                                  index=X_test.index, columns=X_test.columns)
    return X_train_scaling, X_test_scaling
```

```
X_train_scaling, X_test_scaling = feature_engineering_scaling(scaler, X_train, X_test)

X_train_scaling = X_train_scaling[['전류평균', '온도', '선간전압평균', '누적전력량', '주파수', '전압고조파평균', '역률평균']]
X_test_scaling = X_test_scaling[['전류평균', '온도', '선간전압평균', '누적전력량', '주파수', '전압고조파평균', '역률평균']]
y_train = X_train_scaling[['역률평균']]
y_test = X_test_scaling[['역률평균']]
X_train = X_train_scaling[['전류평균', '온도', '선간전압평균', '누적전력량', '주파수', '전압고조파평균']]
X_test = X_test_scaling[['전류평균', '온도', '선간전압평균', '누적전력량', '주파수', '전압고조파평균']]
```

06 사용한 코드

* 계절성 제거(차분)

```
seasonal_lag = 6

for i in X_train_scaling.columns:
    X_train_scaling[i]=X_train_scaling[i].diff(seasonal_lag)

for i in X_test_scaling.columns:
    X_test_scaling[i]=X_test_scaling[i].diff(seasonal_lag)

X_train_scaling = X_train_scaling.dropna()
X_test_scaling = X_test_scaling.dropna()

y_train = X_train_scaling[['역률평균']]
y_test = X_test_scaling[['역률평균']]
X_train = X_train_scaling[['전류평균', '온도', '선간전압평균', '누적전력량', '주파수', '전압고조파평균']]
X_test = X_test_scaling[['전류평균', '온도', '선간전압평균', '누적전력량', '주파수', '전압고조파평균']]
```

* VIF 다중공선성 제거

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

# correlation from features
X_train_scaling.corr().style.background_gradient().set_precision(2).set_properties(**{'font-size': '11pt'})
```

```
# extract effective features using variance inflation factor
vif = pd.DataFrame()

# variance_inflation_factor(X, i) : Xi를 x나머지로 회귀분석한 후 VIF값을 구한것.
# 즉 xi의 vif값. 즉 이값이 높을수록 종속성이 높다는 뜻

vif['VIF_Factor'] = [variance_inflation_factor(X_train_scaling.values, i)
                     for i in range(X_train_scaling.shape[1])]
vif['Feature'] = X_train_scaling.columns
vif.sort_values(by='VIF_Factor', ascending=True)
```

06 사용한 코드

* EDA - 설비 당 기계 1개 추출

```
import pandas as pd

data=pd.read_csv('공조설비_선박엔진.csv')
data['facility_name'].value_counts()
```

```
import matplotlib.font_manager as fm

font_fname = 'C:/Windows/Fonts/Malgun.ttf' #적용할 폰트
font_family = fm.FontProperties(fname=font_fname).get_name() #폰트 설정
plt.rcParams["font.family"] = font_family  #폰트 적용
```

```
stacked1 = data.groupby(["facility_name","역률평균_LABEL"]).size().unstack()
stacked1=stacked1[['정상','주의','경고']]
```

```
stacked1.plot(kind='barh', stacked=True,color=['green','orange','red'])
plt.xticks(rotation=0, fontsize=13)
plt.title('공조설비 설비별 역률평균_LABEL')
plt.xlim([0,4000000])
plt.legend()
plt.show()
```

* EDA - 설비 기계별 변수들의 분포 시각화

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import statsmodels.api as sm

import matplotlib
matplotlib.rcParams['axes.unicode_minus'] = False

import matplotlib.font_manager as fm

font_fname = 'C:/Windows/Fonts/Malgun.ttf' #적용할 폰트
font_family = fm.FontProperties(fname=font_fname).get_name() #폰트 설정
plt.rcParams["font.family"] = font_family  #폰트 적용

raw_data = pd.read_csv("comp1_concat.csv")
comp_1_one = raw_data.drop(columns=['Unnamed: 0'], axis=1)

comp_1_one['NEWTIME'] = pd.to_datetime(comp_1_one['TIME'])
comp_1_one.drop('TIME', axis = 1, inplace=True)
comp_1_one.set_index('NEWTIME', inplace=True)
```

```
X = comp_1_one
X_train = comp_1_one[:43135]
X_test = comp_1_one[43135:]

X_train = X_train[['전류평균','온도', '선간전압평균', '누적전력량',  '주파수', '전압고조파평균','역률평균']]
```

```
plt.style.use('ggplot')
X_train_scaling.plot(figsize=(16,10))
```

```
#따로 시각화
plt.style.use('ggplot')

fig=plt.figure(figsize=(20,20))
ax1=fig.add_subplot(7,1,1)
ax2=fig.add_subplot(7,1,2)
ax3=fig.add_subplot(7,1,3)
ax4=fig.add_subplot(7,1,4)
ax5=fig.add_subplot(7,1,5)
ax6=fig.add_subplot(7,1,6)
ax7=fig.add_subplot(7,1,7)


ax1.plot(X_train_scaling.index,X_train_scaling.전류평균,color='pink',label='전류평균')
ax2.plot(X_train_scaling.index,X_train_scaling.온도,color='orange',label='온도')
ax3.plot(X_train_scaling.index,X_train_scaling.선간전압평균,color='yellow',label='선간전압평균')
ax4.plot(X_train_scaling.index,X_train_scaling.누적전력량,color='green',label='누적전력량')
ax5.plot(X_train_scaling.index,X_train_scaling.주파수,color='blue',label='주파수')
ax6.plot(X_train_scaling.index,X_train_scaling.전압고조파평균,color='purple',label='전압고조파평균')
ax7.plot(X_train_scaling.index,X_train_scaling.역률평균,color='red',label='역률평균')

ax1.legend()
ax2.legend()
ax3.legend()
ax4.legend()
ax5.legend()
ax6.legend()
ax7.legend()
plt.show()
```

06 사용한 코드

* ADF 검정, KPSS 검정

```
import pandas as pd
from statsmodels import datasets
import matplotlib.pyplot as plt
import statsmodels.api as sm
%reload_ext autoreload
%autoreload 2
```

```
from statsmodels.tsa.stattools import adfuller

def adf_test(df):
    result=adfuller(df.values)
    print('ADF Statistic : %f ' % result[0])
    print('p-value : %f' % result[1])
    print('Critical Values:')

    for key,value in result[4].items():
        print('₩t%s : %.3f' % (key,value))

for col in X_train_scaling.columns:
    print(col)
    adf_test(X_train_scaling[col])
```

```
from statsmodels.tsa.stattools import kpss

def kpss_test(df):
    statistic,p_value,n_lags,critical_values=kpss(df.values)

    print(f'KPSS Statistic : {statistic}')
    print(f'p-vluae : {p_value}')
    print(f'num_lags : {n_lags}')
    print('Critical Values : ')

    for key, value in critical_values.items():
        print(f' {key} : {value}')

for col in X_train_scaling.columns:
    print(col)
    kpss_test(X_train_scaling[col])
```

```
import statsmodels.api as sm

for i in X_train_scaling.columns:
    j=1
    fig=plt.figure(figsize=(20,10))
    ax=fig.add_subplot(4,2,j)
    fig=sm.graphics.tsa.plot_acf(X_train_scaling[i],lags=50, ax=ax, use_vlines=True, title='{} ACF'.format(i))
    j+=1
```

06 사용한 코드

* OLS

```
# LinearRegression (using statsmodels)
fit_reg1 = sm.OLS(Y_train, X_train).fit() #회귀분석 계산
display(fit_reg1.summary()) #통계량 정리
```

```
# 예측값
pred_tr_reg1 = fit_reg1.predict(X_train1).values
pred_te_reg1 = fit_reg1.predict(X_test1).values
```

```
def evaluation(Y_real, Y_pred, graph_on=False):
    loss_length = len(Y_real.values.flatten()) - len(Y_pred)
    if loss_length != 0:
        Y_real = Y_real[loss_length:]
    if graph_on == True:
        pd.concat([Y_real, pd.DataFrame(Y_pred, index=Y_real.index, columns=['prediction'])], axis=1).plot(
            kind='line', figsize=(20,6),xlim=(Y_real.index.min(),Y_real.index.max()),linewidth=3, fontsize=20)
        plt.title('Time Series of Target', fontsize=20)
        plt.xlabel('Index', fontsize=15)
        plt.ylabel('Target Value', fontsize=15)
    MAE = abs(Y_real.values.flatten() - Y_pred).mean()
    MSE = ((Y_real.values.flatten() - Y_pred)**2).mean()
    MAPE = (abs(Y_real.values.flatten() - Y_pred)/Y_real.values.flatten()*100).mean()
    Score = pd.DataFrame([MAE, MSE, MAPE], index=['MAE', 'MSE', 'MAPE'], columns=['Score']).T
    Residual = pd.DataFrame(Y_real.values.flatten() - Y_pred, index=Y_real.index, columns=['Error'])
    return Score, Residual
```

```
def evaluation_trte(Y_real_tr, Y_pred_tr, Y_real_te, Y_pred_te, graph_on=False):
    Score_tr, Residual_tr = evaluation(Y_real_tr, Y_pred_tr, graph_on=graph_on)
    Score_te, Residual_te = evaluation(Y_real_te, Y_pred_te, graph_on=graph_on)
    Score_trte = pd.concat([Score_tr, Score_te], axis=0)
    Score_trte.index = ['Train', 'Test']
    return Score_trte, Residual_tr, Residual_te
```

```
# 예측값 평가
Score_reg1, Resid_tr_reg1, Resid_te_reg1 = evaluation_trte(Y_train1, pred_tr_reg1, Y_test1, pred_te_reg1, graph_on=True)
print(Score_reg1)
print(Resid_tr_reg1)
print(Resid_te_reg1)
```

06 사용한 코드

* LSTM

```
comp_1_one=pd.concat([X_train_scaling,X_test_scaling],axis=0)
comp_1_one = comp_1_one.values
```

```
def multivariate_data(dataset, target, start_index, end_index, history_size, target_size, step, single_step=False):
    data = []
    labels = []

    start_index = start_index + history_size
    if end_index is None:
        end_index = len(dataset) - target_size

    for i in range(start_index, end_index):
        indices = range(i - history_size, i, step)
        data.append(dataset[indices])

        if single_step:
            labels.append(target[i + target_size])
        else:
            labels.append(target[i:i + target_size])
    return np.array(data), np.array(labels)
```

```
TRAIN_SPLIT=53274

past_history = 720
future_target = 72
STEP = 6

x_train_multi, y_train_multi = multivariate_data(comp_1_one, comp_1_one[:, -1], 0, TRAIN_SPLIT, past_history,
                                                 future_target, STEP)
x_val_multi, y_val_multi = multivariate_data(comp_1_one, comp_1_one[:, -1], TRAIN_SPLIT, None, past_history,
                                             future_target, STEP)

print('Single window of past history : {}'.format(x_train_multi[0].shape))
print('₩n Target 역률평균 to predict : {}'.format(y_train_multi[0].shape))
```

06 사용한 코드

* LSTM

```
BUFFER_SIZE = 1000
BATCH_SIZE = 1000
EPOCHS = 50
EVALUATION_INTERVAL = 10
```

```
train_data_multi = tf.data.Dataset.from_tensor_slices((x_train_multi, y_train_multi))
train_data_multi = train_data_multi.cache().shuffle(BUFFER_SIZE).batch(BATCH_SIZE).repeat()

val_data_multi = tf.data.Dataset.from_tensor_slices((x_val_multi, y_val_multi))
val_data_multi = val_data_multi.batch(BATCH_SIZE).repeat()
```

```
multi_step_model = tf.keras.models.Sequential()
multi_step_model.add(tf.keras.layers.LSTM(32, return_sequences=True,input_shape=x_train_multi.shape[-2:]))
multi_step_model.add(tf.keras.layers.LSTM(16, activation='relu'))
multi_step_model.add(tf.keras.layers.Dropout(0.3))
multi_step_model.add(tf.keras.layers.Dense(72))

multi_step_model.compile(optimizer='adam', loss='mean_squared_error',metrics=['mae'])
```

```
multi_step_history = multi_step_model.fit(train_data_multi, epochs=EPOCHS,
                                          steps_per_epoch=EVALUATION_INTERVAL,
                                          validation_data=val_data_multi,
                                          validation_steps=50)
```

```
for x, y in val_data_multi.take(1):
    print(multi_step_model.predict(x).shape)
```

06 사용한 코드

* LSTM

```
def plot_train_history(history, title):
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    epochs = range(len(loss))

    plt.figure()

    plt.plot(epochs, loss, 'b', label='Training loss')
    plt.plot(epochs, val_loss, 'r', label='Validation loss')
    plt.title(title)
    plt.legend()

    plt.show()

plot_train_history(multi_step_history,
                   'Multi-Step Training and Validation Loss')
```

```
def multi_step_plot(history, true_future, prediction):
    plt.figure(figsize=(12,6))
    num_in = create_time_steps(len(history))
    num_out = len(true_future)

    plt.plot(num_in, np.array(history[:,-1]), label='History')
    plt.plot(np.arange(num_out)/STEP, np.array(true_future),'b-',label='True Future')

    if prediction.any():
        plt.plot(np.arange(num_out)/STEP,np.array(prediction),'r-',label='Predicted Future')

    plt.legend(loc='upper left')
    plt.xlim([-15,15])
    plt.show()
```

```
def create_time_steps(length):
    return list(range(-length,0))
```

```
for x, y in val_data_multi.take(20):
    multi_step_plot(x[0],y[0],multi_step_model.predict(x)[0])
```

```
pd.DataFrame(y_val_multi).describe()
```

```
loss_MSE,MAE,acc =multi_step_model.evaluate(x_val_multi, y_val_multi, batch_size=1000)

print('loss_MSE : ', loss_MSE)
print('MAE : ', MAE)
```

```
from sklearn.metrics import mean_squared_error, r2_score

y_pred = multi_step_model.predict(x_val_multi)

RMSE=np.sqrt(mean_squared_error(y_val_multi, y_pred))
r2_score = r2_score(y_val_multi, y_pred)
```

06 사용한 코드

* AutoML

```
from pycaret.regression import *

clf = setup(data = X_train_scaling, target = '역률평균', train_size = 0.8, session_id = 6)

best5models = compare_models(sort = 'RMSE', n_select = 5)

best5models
```

```
gbr = create_model('gbr', fold = 5, cross_validation = True)
et = create_model('et', fold = 5, cross_validation = True)
rf = create_model('rf', fold = 5, cross_validation = True)
xgb = create_model('xgboost', fold = 5, cross_validation = True)
knn = create_model('knn', fold = 5, cross_validation = True)
```

```
plot_model(gbr, plot='error')

plot_model(rf, plot='error')
```

```
final_model = finalize_model(gbr)
pred = predict_model(final_model, data = X_test)

plt.plot(X_test_scaling.index, X_test_scaling['역률평균'],color = 'blue')
plt.plot(X_test_scaling.index, pred['Label'], color = 'red')
```

```
final_model = finalize_model(rf)
pred = predict_model(final_model, data = X_test)

plt.plot(X_test_scaling.index, X_test_scaling['역률평균'],color = 'blue')
plt.plot(X_test_scaling.index, pred['Label'], color = 'red')
```

```
# 모델 블렌딩

blender_specific = blend_models(estimator_list = [gbr,rf,xgb], optimize = 'RMSE')
```


감사합니다