2018 발전산업 빅데이터 분석
AI 활용 공모전

과제 분석 결과보고서

1.	**참여 과제**	과제1
2.	**참 가 자**	<team902/ 6명>

주최 | 한국중부발전주식회사 주관 | 한국중부발전주식회사 Microsoft 충남창조경제혁신센터 후원 | DS TRADE KORAIA 지능정보산업협회

1. 데이터 처리과정

데이터 처리 (DNN제외)

정제(Cleansing)

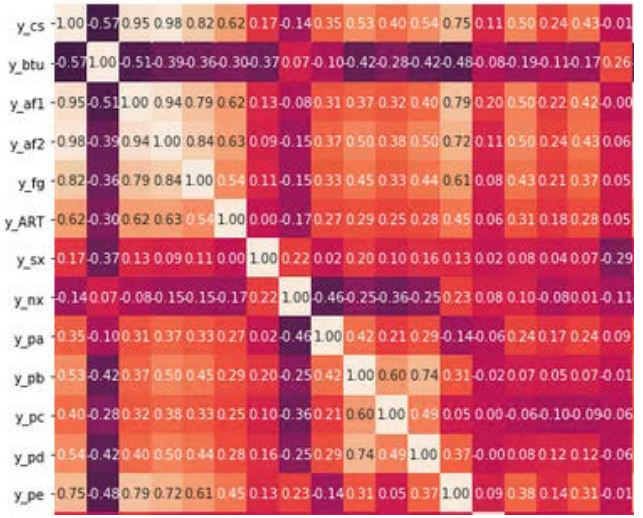
- Re Heater Finish Inlet Steam Temperature Average RH TT01B M XQ03(삭제)->Final ReHeater A Inlet Steam 동일
- Test set에 없는 변수 삭제 -> Coal Feeder A~F SPEED Average CS(6개)
- 석탄성상데이터는 변수에 큰 영향을 주지 않는 것으로 판단하여 사용하지 않음

데이터 이해를 위한 탐색과 시각화

- 상관관계조사

  y변수와 다른 특성 사이의 상관관계 크기가 얼마인지 확인

  →비선형관계의 변수도 있다고 생각되어 모든 변수 사용

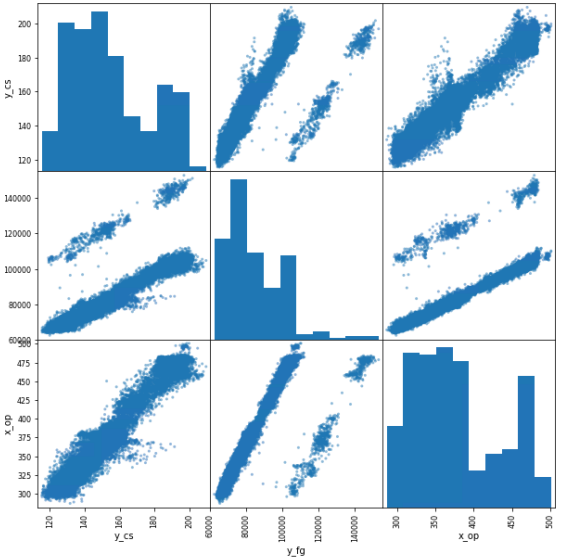


x_co	-0.890746	x_4a	0.744228
x_8b	-0.846580	y_pe	0.746269
x_9b	-0.845544	x_na	0.747669
x_8a	-0.808869	x_wsh2	0.800601
y_btu	-0.568913	y_fg	0.822534
x_9a	-0.316615	x_21	0.849637
x_MB	-0.243902	x_FA	0.861011
y_nx	-0.144076	x_12	0.935318
x_7a	-0.076976	x_11	0.936350
x_cn	-0.034665	y_af1	0.946822
y_PCLINK	-0.010470	x_op	0.967345
x_cs	0.072754	y_af2	0.976150
x_7b	0.094889	y_cs	1.000000

- 산점도 행렬을 이용한 시각화

  두 변수 사이의 산점도 확인 결과 특정 조합에서 두 개의 다른 분포가 보임

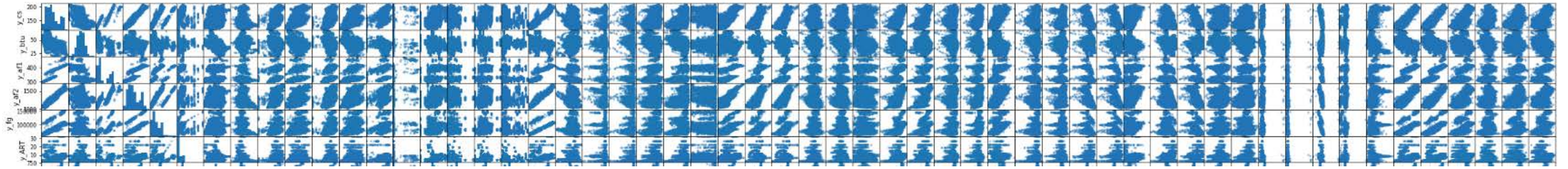
  → 두 분포의 데이터를 분리하여 분석(y_fg, y_cs)



1. 데이터 처리과정

데이터 EDA (DNN)

1. Correlation matrix를 통한 데이터간 상관성 시각화
   - 이상치 탐색



   - N/A 및 #ref 이상치는 엑셀을 통해 제거

2. 변수 별 Correlation 값 확인
   - 상관계수가 0.9 이상 되는 변수는 두 변수 중 하나 삭제
   - 탈질설비 B INLET GAS TEMPERATURE, A/B Platen SuperHeater Outlet Temperature Average, ReHeater DeSuperHeater Inlet Temperature, ReHeater B DeSuperHeater Outlet Temperature Average, Second Economizer Outlet LINK Steam Temperature

```
abs(fulldata.corr()).x_8a.sort_values(ascending=False).head(20)
```

```
x_8a      1.000000
x_8b      0.991494
x_9b      0.873387
y_af2     0.836892
```

2. 분석과정 및 결과

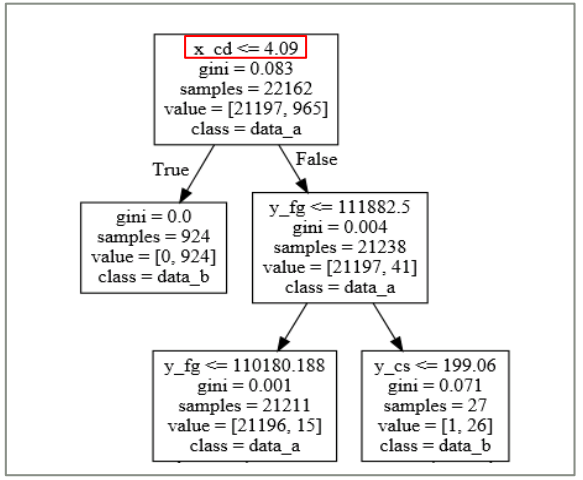
머신러닝 알고리즘을 위한 데이터 준비(DNN제외)

1) Clustering

'Flue Gas'와'급탄량'의 산점도에서 분포가 두 개로 나뉘어 지는 것을 확인하였다. DBSCAN을 사용하여 train set을 Class 1과 Class 0으로 군집화. 정확한 예측 결과 도출을 위해 두 부분으로 나누어 분석을 진행함.

2) Decision Tree

Test 셋의 분리를 위해 train set의 클러스터 기준을 찾기 위해 decision tree를 이용해 중요 변수 기준을 도출함.



이 결과에 의하여 Test Set을 '굴뚝 DUST'의 4.09를 기준으로 분리

데이터 준비(DNN제외)

1. y_data와 x_data로 데이터 셋 구성
2. Train set와 val set, test set(제출)로 구성
3. MinMaxScaler로 x_train, val, test 특성 스케일링

```
x_data = pd.DataFrame(data_a_check.filter(regex='x_'))
#x_data = x_data.iloc[:,list(np.arange(0,38))]
y_data = pd.DataFrame(data_a_check['y_cs'])
data = pd.concat([x_data, y_data], axis=1)
```

```
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_data, y_data, random_state = 0)
```

```
print(x_train.shape, x_val.shape, y_train.shape, y_val.shape)
```

```
(15897, 38) (5300, 38) (15897, 1) (5300, 1)
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(x_train)
#x_scaled = scaler.transform(x_data)
x_train_scaled = scaler.transform(x_train)
x_val_scaled = scaler.transform(x_val)
x_test = scaler.transform(real_a)
```

2. 분석과정 및 결과

딥러닝 알고리즘을 위한 데이터 준비(DNN)

1. 학습 데이터 분리
   - Train : Validation : Test (7 : 1.5 : 1.5)
   - Split끼리 random_state를 고정하여 데이터 혼선 방지

```
xtr, xvalts, ytr, yvalts = train_test_split(train_x, train_y, random_state = 33, test_size=0.3)
xval, xts, yval, yts = train_test_split(xvalts, yvalts, random_state = 33, test_size=0.5)
```

2. 2차 예측을 위한 2차 test 데이터 준비

```
## 2차 y값 도출을 위한  test data
test2_del_vars = test[['x_op', 'x_WF', 'x_7a', 'x_7b', 'x_11a', 'x_11b', 'x_MB', 'x_cfa', 'x_cfb', 'x_cfc', 'x_cfd', 'x_cfe', 'x_cff', 'x_FA', 'x_na', 'x_3a', 'x_3b', 'x_wsh1',
                       'x_4a', 'x_4b', 'x_wsh2', 'x_6a', 'x_6b', 'x_8b', 'x_wrh1', 'x_9a', 'x_10a', 'x_10b', 'x_cs', 'x_cn', 'x_co', 'x_cd', 'x_ash', 'x_11', 'x_21', 'x_24', 'x_22', 'x_23']]
```

3. 데이터 스케일링
   - Min Max 스케일러 사용
   - Train 데이터 스케일 fit을 validation 과 test, 그리고 2차 test 데이터에도 적용

```
scaler = MinMaxScaler()
scaler.fit(xtr)
xtr_scaled = pd.DataFrame(scaler.transform(xtr))
xval_scaled = pd.DataFrame(scaler.transform(xval))
xts_scaled = pd.DataFrame(scaler.transform(xts))
ts_scaled = pd.DataFrame(scaler.transform(test2_del_vars))
```

2. 분석과정 및 결과

모델 훈련 예시) 급탄량(y_cs) 모델링

Linear Regression

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression().fit(x_train_scaled, y_train)
lr2 = LinearRegression().fit(x_train_scaled_2, y_train_2)
```

Elastic Net

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import ElasticNet

param_grid = {'alpha':[0.001,0.01,0.1,1,10,100], 'l1_ratio':[0.1,0.3,0.5,0.7,1]}
grid_search = GridSearchCV(ElasticNet(), param_grid, cv = 5)
grid_search.fit(x_train_scaled, y_train)
grid_search2 = GridSearchCV(ElasticNet(), param_grid, cv = 5)
grid_search2.fit(x_train_scaled_2, y_train_2)

enet_1 = ElasticNet(alpha = 0.001, l1_ratio = 1).fit(x_train_scaled, y_train)
enet_2 = ElasticNet(alpha = 0.001, l1_ratio =1).fit(x_train_scaled_2, y_train_2)
```

→그리드 서치를 이용하여 최적 파라미터 (alpha =0.001, l1_ratio=1) 구하고, 이를 모델링에 사용

LinearSVR

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import LinearSVR
param_grid = {'C':[0.001,0.01,0.1,1,10,100],'epsilon':[0.001,0.01,0.1,1,10,100]}

grid_search = GridSearchCV(LinearSVR(), param_grid, cv = 5)
grid_search.fit(x_train_scaled, y_train)
grid_search2 = GridSearchCV(LinearSVR(), param_grid, cv = 5)
grid_search2.fit(x_train_scaled_2, y_train_2)

svm_1 = LinearSVR(C = 10, epsilon = 0.001).fit(x_train_scaled, y_train)
svm_2 = LinearSVR(C = 100, epsilon = 1).fit(x_train_scaled_2, y_train_2)
```

→그리드 서치를 이용하여 최적 파라미터 (C=10, epsilon=0.001) 구하고, 이를 모델링에 사용

XGBoost Regressor

```
from sklearn.model_selection import GridSearchCV
import xgboost as xgb

param_grid = {'n_estimators':[100,1000], 'learning_rate' : [0.1,0.2,0.5,0.7,1.0]}
grid_search = GridSearchCV(xgb.XGBRegressor(), param_grid, cv = 3)
grid_search.fit(x_train_scaled, y_train)

xgb_1 = xgb.XGBRegressor(learning_rate=0.1, n_estimators=1000).fit(x_train_scaled, y_train)
xgb_2 = xgb.XGBRegressor(learning_rate=0.1, n_estimators=1000).fit(x_train_scaled_2, y_train_2)
```

→그리드 서치를 이용하여 최적 파라미터 (learning_rate = 0.1, n_estimators =1000) 구하고, 이를 모델링에 사용

2. 분석과정 및 결과

모델 훈련

DNN(케라스 사용)

- 3개의 모델 MSE 및 앙상블 모델의 MSE 계산
- 앙상블 모델의 경우,

  종속변수마다 한 개, 두 개, 또는 세 개의 모델의 조합으로 구성

```
cs1 = Sequential()
cs1.add(Dense(128, input_dim=38, activation = 'relu'))
for i in range(3):
    cs1.add(Dense(64, activation ='relu'))
cs1.add(Dense(1))

cs1.compile(optimizer = 'rmsprop', loss='mse')
early_stopping = EarlyStopping(patience = 50)
hist1 = cs1.fit(xtr_scaled, ytr_cs, batch_size = 256, epochs=500,
                validation_data = (xval_scaled, yval_cs), callbacks=[early_stopping], verbose=0)
print(cs1.evaluate(xts_scaled, yts_cs, batch_size=256))
pd.DataFrame(cs1.predict(xts_scaled)).plot()
yts_cs.reset_index().y_cs.plot()
```

```
cs2 = Sequential()
cs2.add(Dense(128, input_dim=38, activation = 'relu'))
cs2.add(Dense(64, activation ='relu'))
cs2.add(Dense(64, activation ='relu'))
cs2.add(Dense(1))

cs2.compile(optimizer = 'rmsprop', loss='mse')
early_stopping = EarlyStopping(patience = 50)
hist2 = cs2.fit(xtr_scaled, ytr_cs, batch_size = 256, epochs=500,
                validation_data = (xval_scaled, yval_cs), callbacks=[early_stopping], verbose=0)
print(cs2.evaluate(xts_scaled, yts_cs, batch_size=256))
pd.DataFrame(cs2.predict(xts_scaled)).plot()
yts_cs.reset_index().y_cs.plot()
```

```
print('mse1:',mean_squared_error(cs1.predict(xts_scaled), yts_cs),
      '\nmse2:', mean_squared_error(cs2.predict(xts_scaled), yts_cs), '\nmse3:',mean_squared_error(cs3.predict(xts_scaled), yts_cs),
      '\nmseavg:', mean_squared_error((cs1.predict(xts_scaled)+cs2.predict(xts_scaled)+cs3.predict(xts_scaled))/3, yts_cs))
```

```
mse1: 11.64434680513407
mse2: 7.421575448171412
mse3: 5.918997951978934
mseavg: 1.342505407938304
```

2. 분석과정 및 결과

모델 선택(결과)

MSE 비교(최소 MSE 모델 선택)

	Linear Regression	Elastic Net	SVM Regressor	XGBoost Regressor	ANN
급탄량(석탄사용량)	12.95	13.15	13.1	11.71	1.34
BTU Compensation	19.68	19.91	20.13	7.12	3.28
Primary Air FLOW	54.39	54.61	55.5	18.51	35.76
Secondary Air FLOW	145.23	145.39	146.92	157.11	366.88
Flue GAS FLOW	1188149.29	1187691.67	4368877.63	1231866.25	5068608.17
튜브마모율 %	5.96	5.95	6.41	3.93	1.24
탈황설비 입구 SOx	277.97	277.78	283.25	190.06	215.27
탈질설비 입구 Nox	93.24	93.6	95.75	70.44	22.76
PULVERIZER A BOWL	2347.16	2346.82	2785.61	1015.66	124.03
PULVERIZER B BOWL	504.79	506.14	510.77	372.48	139.48
PULVERIZER C	1952.27	1953.64	2358.66	1172.87	214.04
PULVERIZER D	558.91	560.58	564.52	431.75	162.01
PULVERIZER E	1878.84	1880.15	2036.5	1120.62	83.26
PULVERIZER F	60.44	60.69	64.01	26.36	2.09
미연분Fly Ash (wt%)	1.1	1.13	1.12	0.87	0.16
미연분Bed Ash (wt%)	30.07	30.09	33.92	21.11	2.45
미연분평균 (wt%)	2.53	2.56	2.6	2.05	0.26
클리커예측 지수	0.03	0.03	0.03	0.02	0.00

→ MSE가 가장 낮은 DNN(인공신경망)모델을 사용

2차 test 데이터의 y값 산출

- 앙상블 모델에 2차 test데이터 투입, 최종 y값 계산
- Excel로 결과 값 export

```
y_cs = pd.DataFrame()
y_cs['y_cs1'] = cs1.predict(ts_scaled).flatten()
y_cs['y_cs2'] = cs2.predict(ts_scaled).flatten()
y_cs['y_cs3'] = cs3.predict(ts_scaled).flatten()
y_cs['y_cs_avg'] = (y_cs['y_cs1']+y_cs['y_cs2']+y_cs['y_cs3'])/3
y_cs.to_csv('y_cs.csv', mode='w')
```

3. 시사점 및 제언

- 크로스 벨리데이션과 같은 과적합 방지 모델링을 하였다면 더 좋은 일반화 모델링이 가능할 것으로 예상
- 데이터의 빈구간이 너무 커서 시계열분석에 어려움이 있음
- 이 부분을 보완하여 시계열분석, RNN LSTM과 같은 모델링을 사용한다면 주기성을 보이는 패턴을 찾고 더욱 정확한 예측을 할 수 있음

- End of Document -