Optimizing Process

후판 공정 데이터 분석을 통해 SCALE불량의 원인 도출 및 개선 방안 수립

POSCO AI Bigdata Academy 12기
A반 1조 성유기

과제 정의

분석배경 : 후판공정의 Scale불량 발생 증가

분석목적 : Scale불량 발생 원인 도출 및 개선책 수립

분석방향
- step1. 단변수/이변수 분석을 통해 잠재인자 선정 및 분석 방향성 확인
- Step2. 분류 모델을 통해 scale불량 발생에 영향을 주는 주요인자 도출
- Step3. scale 발생 원인 도출 및 개선 방향 제시

분석목표
- 분석을 통해 불량 발생의 근본 원인을 찾고, 결과를 해석하여 개선기회 도출

A. 데이터 전처리

1-1. 데이터 셋 확인
1-2. RAW 데이터 확인(히스토그램, 교차분석, 상관분석)
2. 결측값처리
3. 이상값 처리
4. Feature Engineering

B. 모델링

1. 분류모델(의사결정나무, 랜덤포레스트)
2. 중요인자 변수 도출

C. 분석 결과 및 결론

1-1. 데이터 셋 확인

	A	B	C	D	E
1	**변수**	**변수 설명**	**변수 역할**	**변수 형태**	**분석O,X**
2	PLATE_NO	Plate Number	ID	범주형	X
3	ROLLING_DATE	작업시각	제외	연속형	X
4	SCALE	Scale불량	목표변수	범주형	O
5	SPEC	제품 규격	설명변수	범주형	X
6	STEEL_KIND	강종	설명변수	범주형	O
7	PT_THK	plate 두께	설명변수	연속형	O
8	PT_WDTH	plate 폭	설명변수	연속형	O
9	PT_LTH	plate 길이	설명변수	연속형	O
10	PT_WGT	plate 중량	설명변수	연속형	O
11	FUR_NO	가열로 호기	설명변수	범주형	O
12	FUR_NO_ROW	가열로 작업 순번	설명변수	연속형	O
13	FUR_HZ_TEMP	가열로 가열대 온도	설명변수	연속형	O
14	FUR_HZ_TIME	가열로 가열대 시간	설명변수	연속형	O
15	FUR_SZ_TEMP	가열로 균열대 온도	설명변수	연속형	O
16	FUR_SZ_TIME	가열로 균열대 시간	설명변수	연속형	O
17	FUR_TIME	가열로 시간	설명변수	연속형	O
18	FUR_EXTEMP	추출 온도	설명변수	연속형	O
19	ROLLING_TEMP_T5	압연 온도	설명변수	연속형	O
20	HSB	HBS 적용	설명변수	범주형	O
21	ROLLING_DESCALING	압연 중 descaling 횟수	설명변수	연속형	O
22	WORK_GR	작업조	설명변수	범주형	O

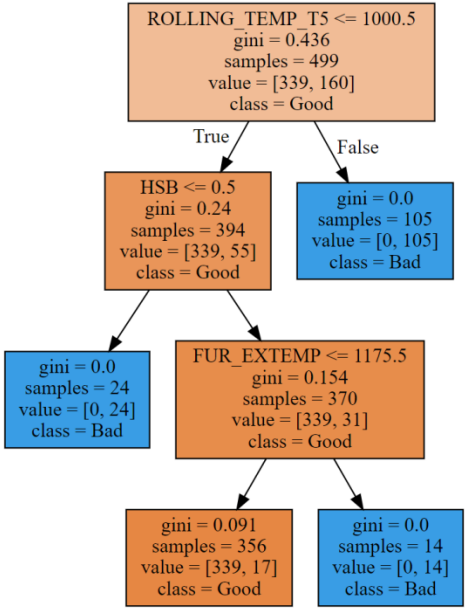
- 설명변수 : 이산형 + 범주형
- 목표변수 : 범주형(양품/불량)
  => 모델링시 분류 모델이용

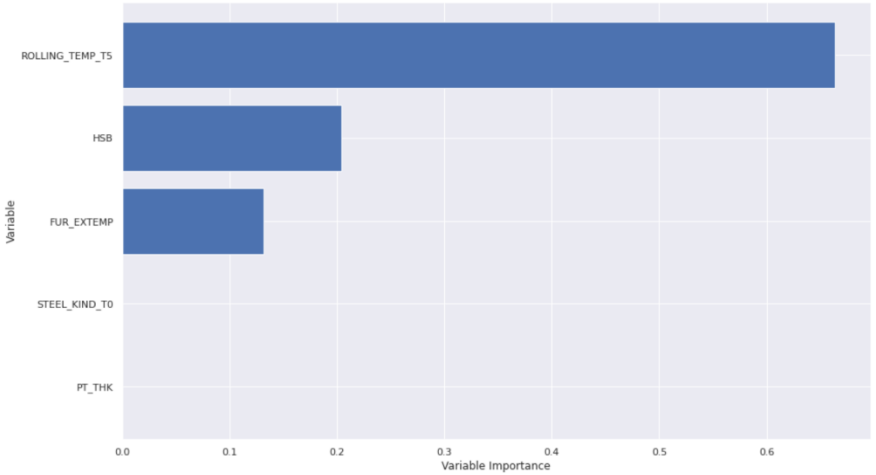
- 연속형 변수 : 히스토그램, 상관분석 (다중공선성 확인)
- 범주형 변수 : 교차분석

1-2. RAW 데이터 확인 후 각 변수 별 분석 계획 수립

변수	변수 설명	변수 역할	변수 형태	분석O,X	분석사유 및 분석제외사유
PLATE_NO	Plate Number	ID	범주형	X	제품 고유 번호와는 무관
ROLLING_DATE	작업시각	제외	연속형	X	작업 시작 시간과는 무관
SCALE	Scale불량	목표변수	범주형	O	목표변수
SPEC	제품 규격	설명변수	범주형	X	특별한 규칙 없음
STEEL_KIND	강종	설명변수	범주형	O	C0강종의 불량 비율 높음
PT_THK	plate 두께	설명변수	연속형	O	두께가 작을수록 불량 많음
PT_WDTH	plate 폭	설명변수	연속형	O	폭이 2000~2500까지 불량률 증가
PT_LTH	plate 길이	설명변수	연속형	O	30000~45000 사이에서 불량률 높음
PT_WGT	plate 중량	설명변수	연속형	O	특별한 규칙 없음
FUR_NO	가열로 호기	설명변수	범주형	O	특별한 규칙 없음
FUR_NO_ROW	가열로 작업 순번	설명변수	연속형	O	특별한 규칙 없음
FUR_HZ_TEMP	가열로 가열대 온도	설명변수	연속형	O	1160도 근처에서 불량률 높고 1200도 이상에서 모두 불량
FUR_HZ_TIME	가열로 가열대 시간	설명변수	연속형	O	특정구간에서 불량률 높음
FUR_SZ_TEMP	가열로 균열대 온도	설명변수	연속형	O	1150~60 구간에서 불량률 높고, 1175도 이상에서 모두 불량
FUR_SZ_TIME	가열로 균열대 시간	설명변수	연속형	O	특정구간에서 불량률 높음
FUR_TIME	가열로 시간	설명변수	연속형	O	특정구간에서 불량률 높음
FUR_EXTEMP	추출 온도	설명변수	연속형	X	fur_sz_temp와 상관관계있음
ROLLING_TEMP_T5	압연 온도	설명변수	연속형	O	특정온도 이상에서만 발생(유력)
HSB	HBS 적용	설명변수	범주형	O	미적용제품의 경우 모두 불량(유력)
ROLLING_DESCALING	압연 중 descaling 횟수	설명변수	연속형	O	특별한 규칙 없음
WORK_GR	작업조	설명변수	범주형	O	2조의 불량률이 낮은편

분류모델(의사결정나무)



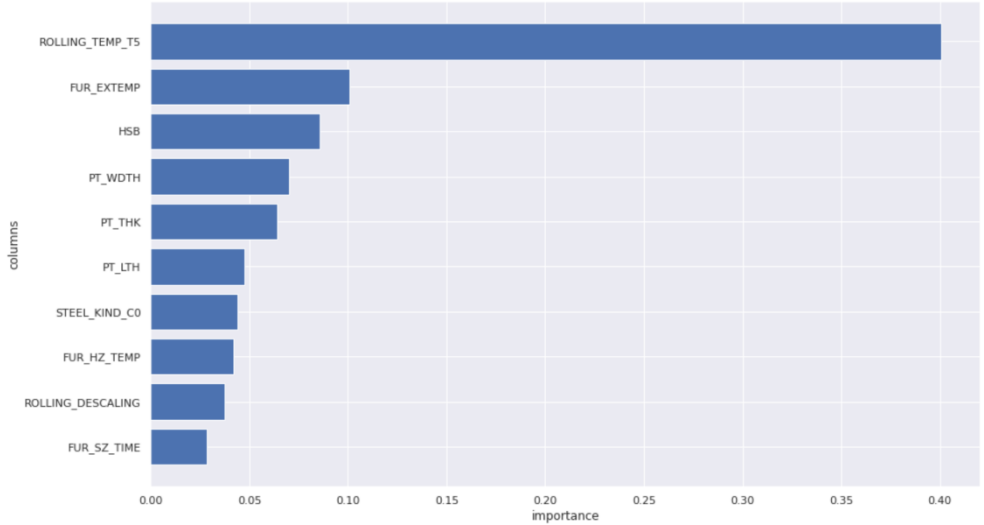


ROLLING_TEMP_T5(압연온도) >> HBS적용 유무 > FUR_EXTEMP(추출온도) 순으로 중요

=> 압연온도가 1000.5 도 보다 높은 경우, HBS를 적용하지 않은 경우, 추출온도가 1175.5 보다 높은 경우 100% 불량 발생

분류모델(랜덤포레스트)

	Feature	Importance
10	ROLLING_TEMP_T5	0.401
9	FUR_EXTEMP	0.101
11	HSB	0.086
1	PT_WDTH	0.070
0	PT_THK	0.064
2	PT_LTH	0.047
13	STEEL_KIND_C0	0.044
5	FUR_HZ_TEMP	0.042
12	ROLLING_DESCALING	0.037
7	FUR_SZ_TIME	0.028



ROLLING_TEMP_T5(압연온도) >> FUR_EXTEMP(추출온도) > HBS적용 유무 순으로 중요

개선방안

1. 압연온도를 1000.5도를 넘기지 않도록 최대 950도 까지로 유지한다.

- 의사 결정 나무와 랜덤포레스트를 통해 scale불량에 영향을 주는 가장 큰 요소는 압연온도이다.
- 의사결정나무에서 첫번째 가지가 나누어 지는 기준을 보면 1000.5도 이상일 경우 105개의 샘플 모두 불량으로 귀결된다.

2. HBS 적용을 해준다.

- HSB 적용을 하는 경우 모두가 scale발생 하지 않음

3. 추출온도를 1175.5 도를 넘기지 않도록 최대 1150 도 정도로 유지한다.

- 추출온도가 1175.5 도 이상일 경우 모두 scale불량이 발생

4. 현업 엔지니어에게 도메인지식을 요청하여 C0강종의 scale불량에 대한 조치를 취한다.

- 모델링에서는 발견되지 않았지만, 교차분석을 통해 C0강종에 대해 불량률이 현저히 높게 나왔다. C0 강종의 특징상 scale이 많이 발생하는 강종인 것으로 파악되는데, 이런 결과가 발생한 이유를 현업 엔지니어와 함께 C0강종의 scale불량에 대한 조치를 취한다.

감사합니다

[첨부]데이터 전처리

1-1. 데이터 셋 확인

```
1 import pandas as pd
2
3 df_raw = pd.read_csv('SCALE불량.csv', encoding='cp949')
4 df_raw.head(3)
```

	PLATE_NO	ROLLING_DATE	SCALE	SPEC	STEEL_KIND	PT_THK	PT_WDTH	PT_LTH	PT_WGT	FUR_NO	FUR_NO_ROW	FUR_HZ_TEMP	FUR_HZ_TIME	FUR_SZ_TEMP	FUR_SZ_TIM
0	PB562774	2008-08-01:00:00:15	양품	AB/EH32-TM	T1	32.25	3707	15109	14180	1호기	1	1144	116	1133	5
1	PB562775	2008-08-01:00:00:16	양품	AB/EH32-TM	T1	32.25	3707	15109	14180	1호기	2	1144	122	1135	5
2	PB562776	2008-08-01:00:00:59	양품	NV-E36-TM	T8	33.27	3619	19181	18130	2호기	1	1129	116	1121	5

[첨부]데이터 전처리

1-1. 데이터 셋 확인

```
1 df_raw.info()
```

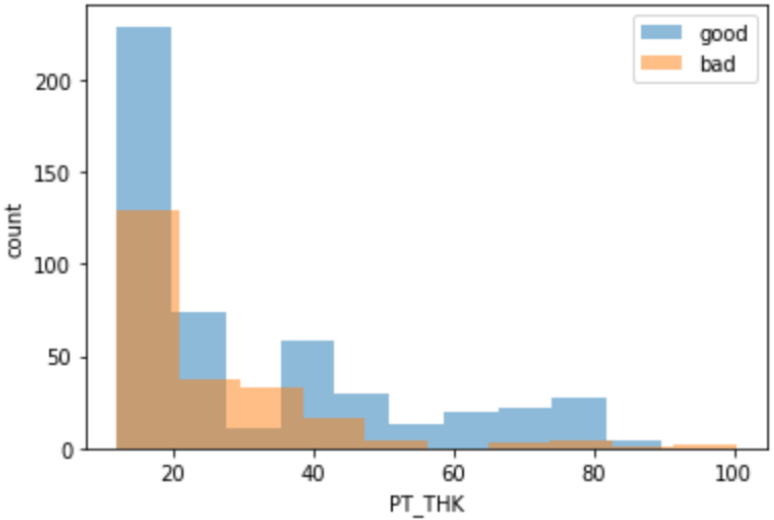
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 720 entries, 0 to 719
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   PLATE_NO           720 non-null    object
 1   ROLLING_DATE       720 non-null    object
 2   SCALE              720 non-null    object
 3   SPEC               720 non-null    object
 4   STEEL_KIND         720 non-null    object
 5   PT_THK             720 non-null    float64
 6   PT_WDTH            720 non-null    int64
 7   PT_LTH             720 non-null    int64
 8   PT_WGT             720 non-null    int64
 9   FUR_NO             720 non-null    object
 10  FUR_NO_ROW         720 non-null    int64
 11  FUR_HZ_TEMP        720 non-null    int64
 12  FUR_HZ_TIME        720 non-null    int64
 13  FUR_SZ_TEMP        720 non-null    int64
 14  FUR_SZ_TIME        720 non-null    int64
 15  FUR_TIME           720 non-null    int64
 16  FUR_EXTEMP         720 non-null    int64
 17  ROLLING_TEMP_T5    720 non-null    int64
 18  HSB                720 non-null    object
 19  ROLLING_DESCALING  720 non-null    int64
 20  WORK_GR            720 non-null    object
dtypes: float64(1), int64(12), object(8)
memory usage: 118.2+ KB
```

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인

```
# 연속형 변수에 대한 양품, 불량 분포 확인

def fun_plot_hist(data, var):
    plt.hist(data[data['SCALE'] == 0][var], label = "good", alpha = 0.5)
    plt.hist(data[data['SCALE'] == 1][var], label = "bad", alpha = 0.5)
    plt.legend()
```

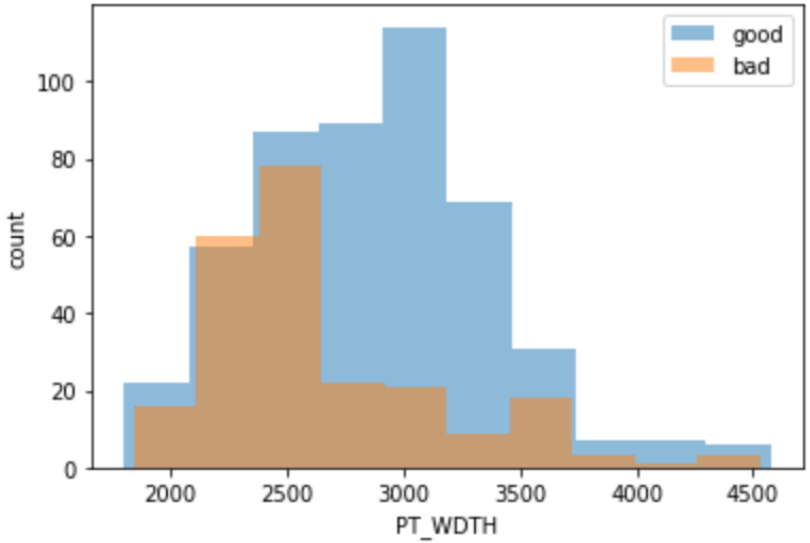
```
# 연속형 변수 목록
numeric_columns_list = ["PT_THK","PT_WDTH","PT_LTH","PT_WGT","FUR_NO_ROW","FUR_HZ_TEMP","FUR_HZ_TIME","FUR_SZ_TEMP","FUR_SZ_TIME","FUR_TIME","FUR_EXTEMP","ROLLING_TEMP_T5","ROLLING_DESCALING","HSB"]

for i in numeric_columns_list :
  fun_plot_hist(df_raw,i)
  plt.xlabel(i)
  plt.ylabel("count")
  plt.show()
```

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인



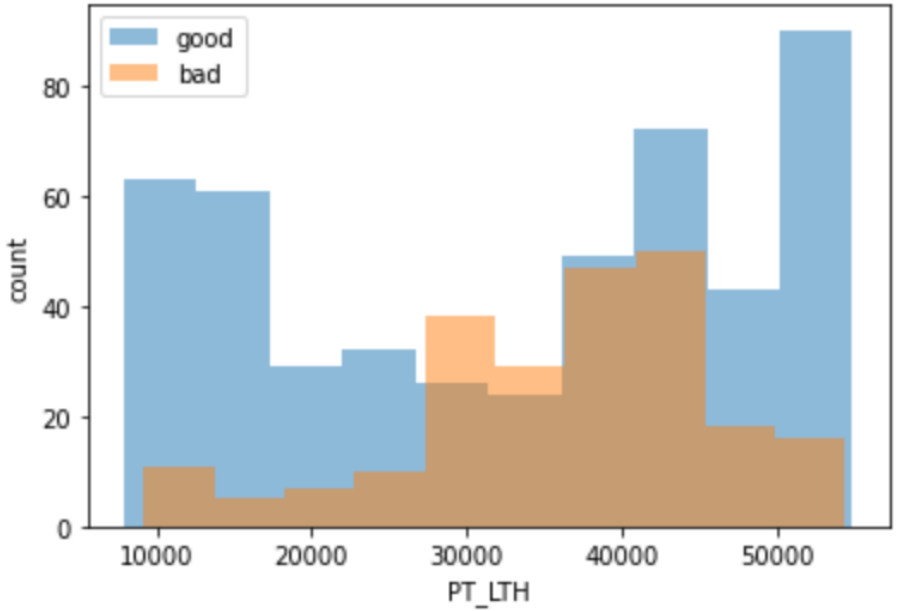
플레이트의 두께 폭이 작을 수록 불량이 많은 편이다.

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인



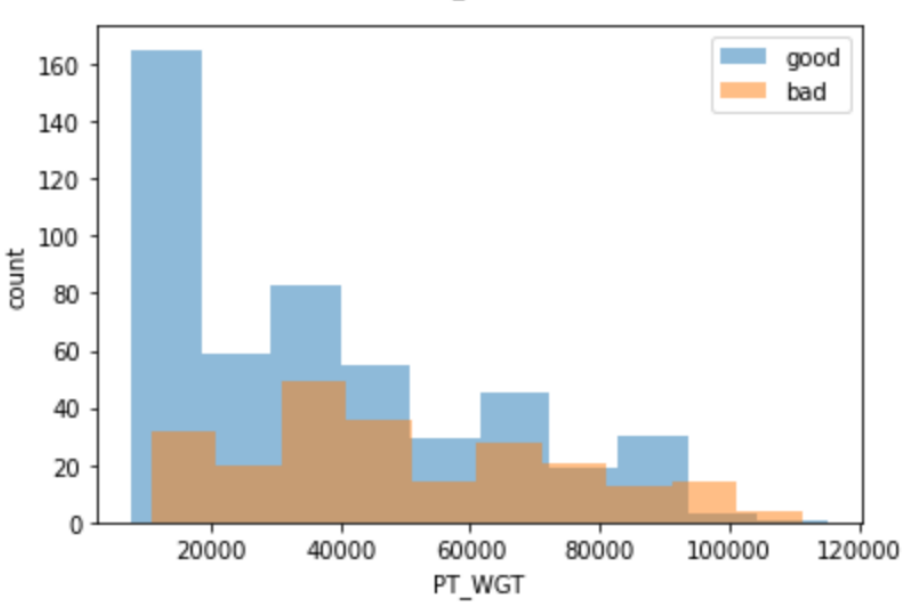
플레이트의 폭이 2500 까지 불량률이 많은 편이다.

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인



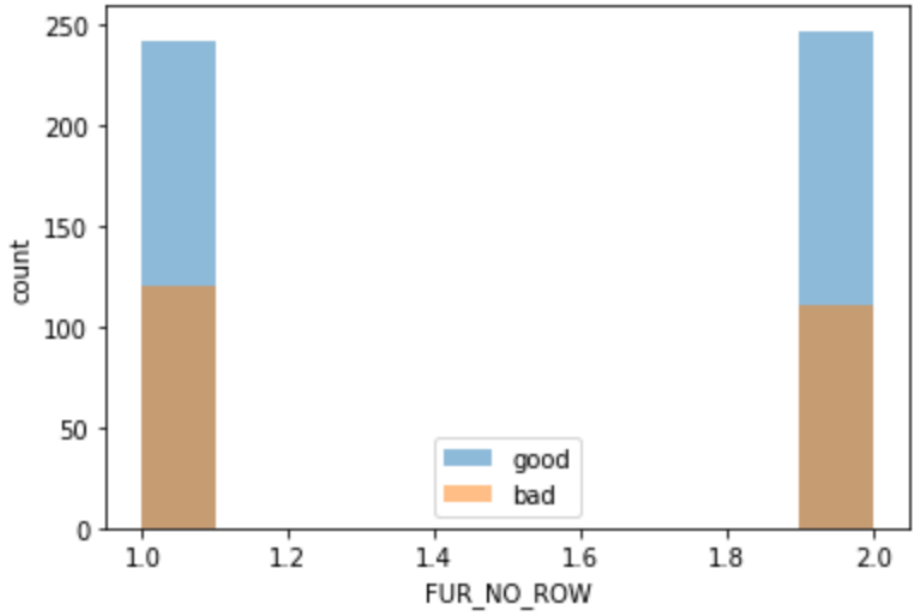
특정 구간에 대한 불량률이 높다

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인

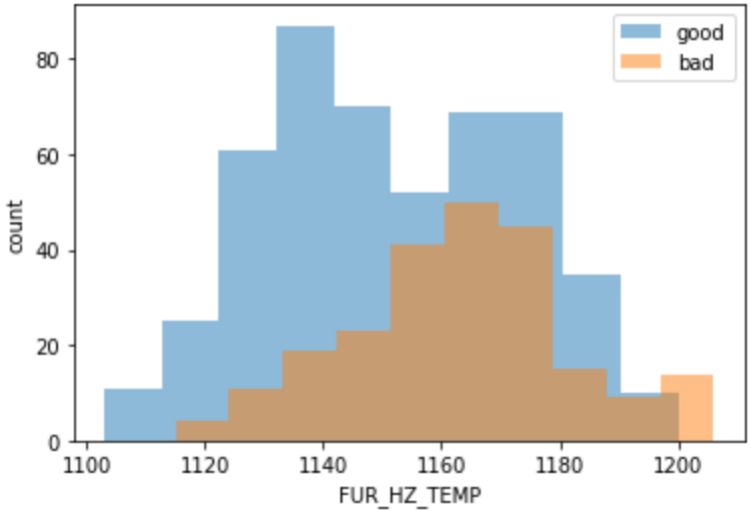


플레이트의 중량에 따라 scale 불량의 비율이 큰 차이가 없다.

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인

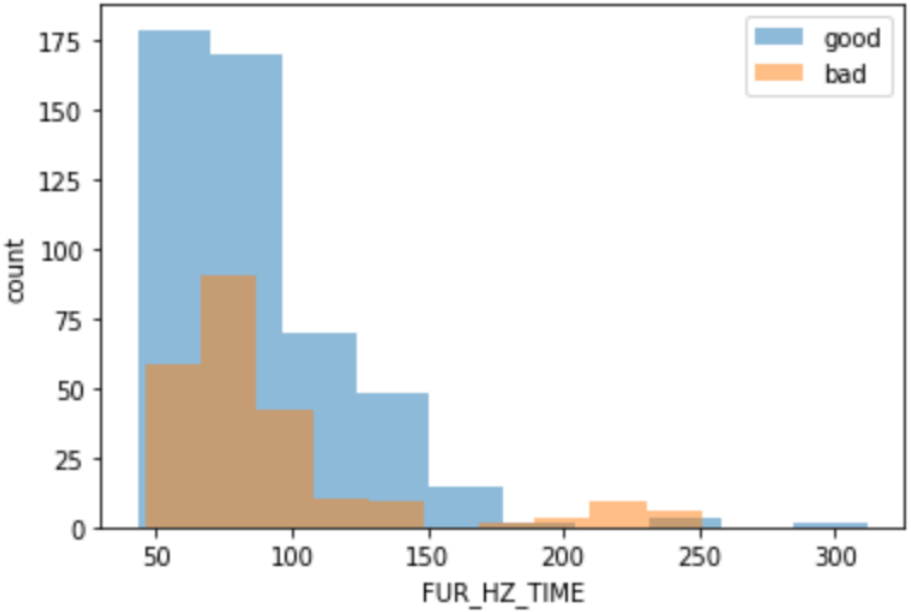


1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인



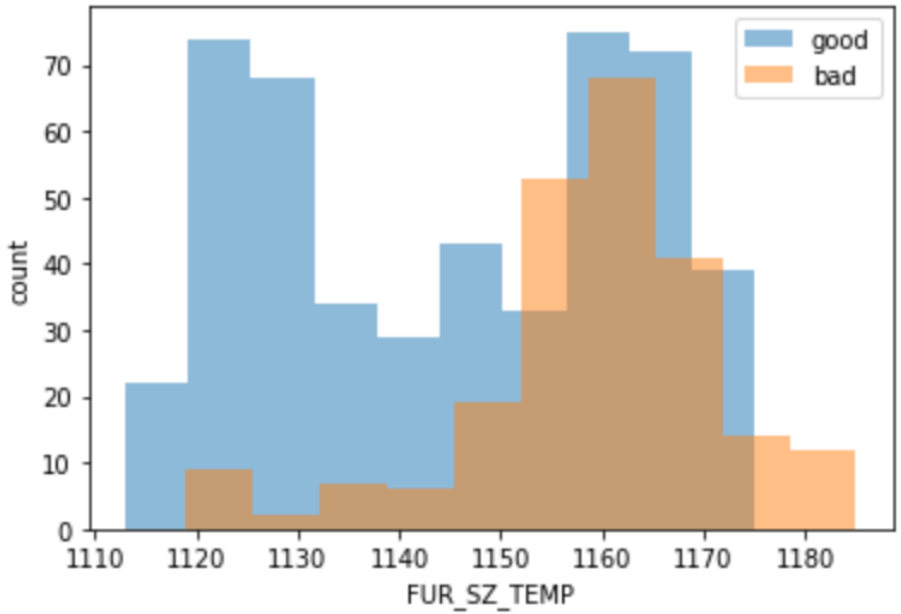
가열로 온도는 1160도 근처에서 불량률이 높고, 1200도 이상에서 모두 불량이다.

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인



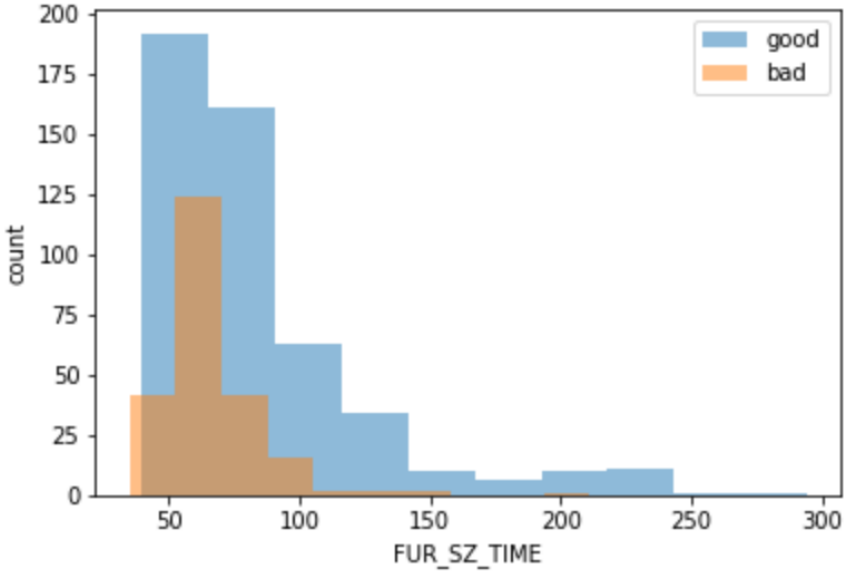
가열로 시간에 따라 scale 불량의 변화가 크지 않음.

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인



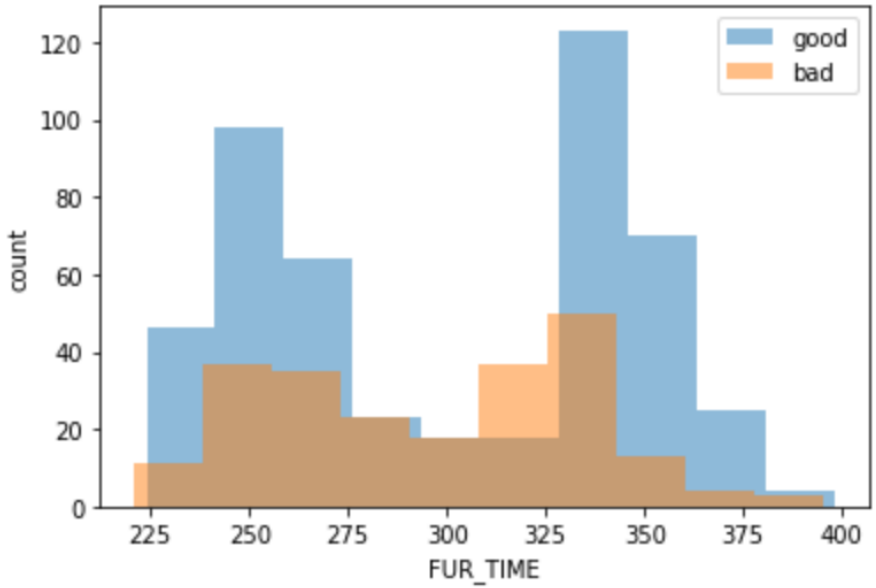
1175도 이상에서는 모두 scale 불량이 발생하는 것을 보아
FUR_SZ_TEMP온도와 불량과 밀접한 관련이 있음을 추측

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인

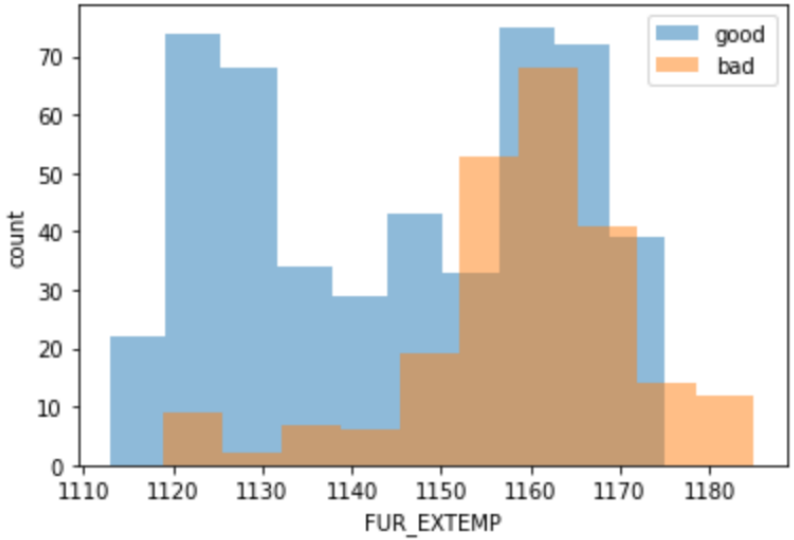


불량률이 특정 구간에서 높다.

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인

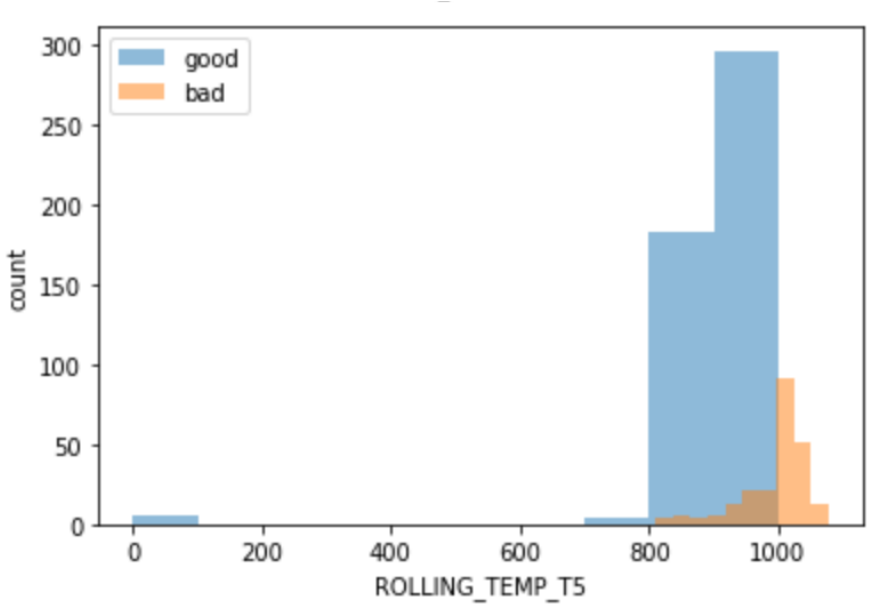


1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인



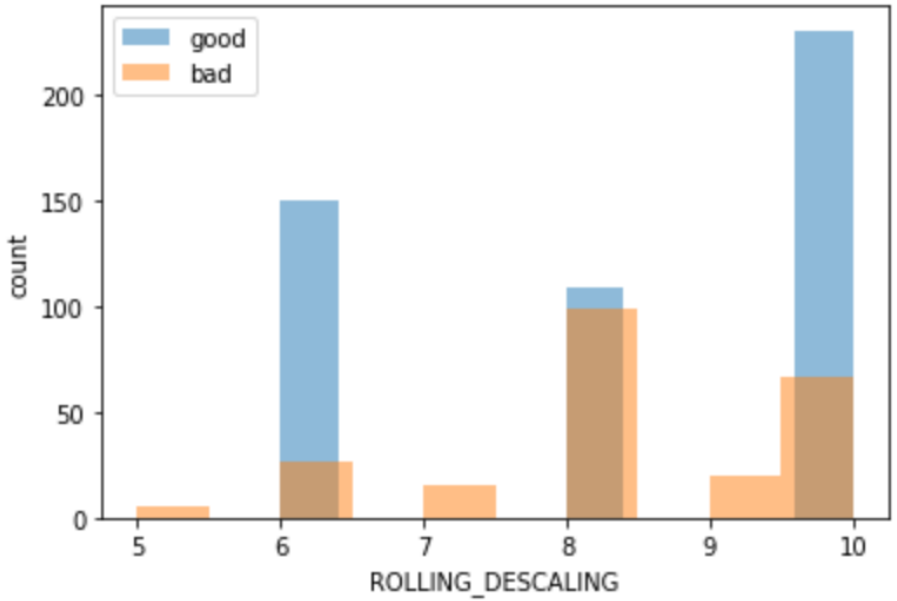
1175도 이상에서는 모두 scale 불량이 발생하는 것을 보아
FUR_EXTEMP온도와 불량과 밀접한 관련이 있음을 추측

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인



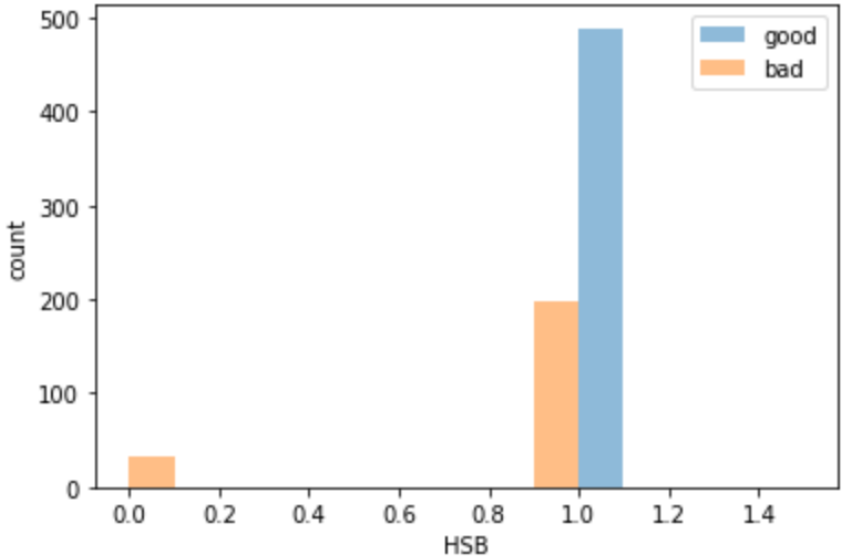
특정 구간(1000도 이상)에서만 불량이 발생하며, 이상치가 존재한다.

1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인



히스토 그램상으로는 큰 변화가 보이지는 않지만, Descaling은 스케일을 제거해주는 작업이므로 스케일 불량에 중요한 요인 일 것으로 추측한다.

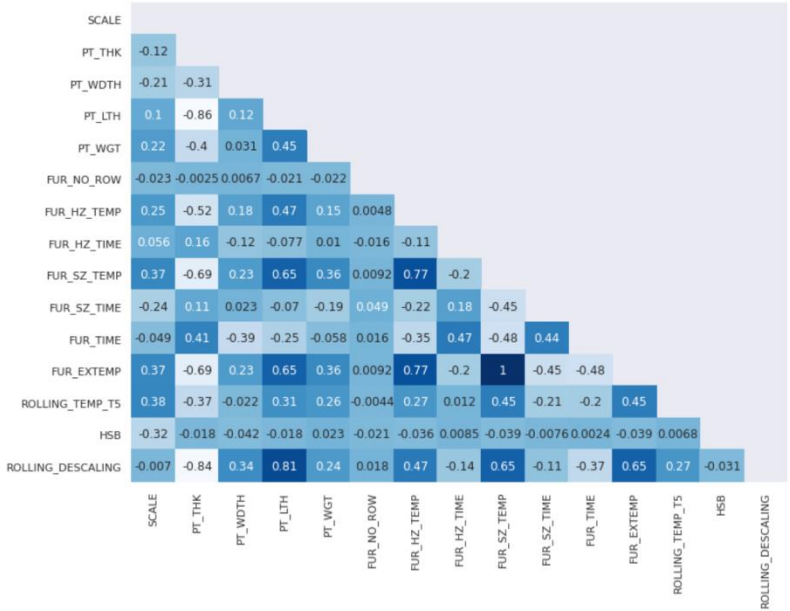
1-2. RAW data확인 - 연속형 변수에 대한 양품, 불량 분포 확인



HSB 적용한 경우는 모두 양품으로 확인된다.

1-2. RAW data확인 - 연속형 변수에 대한 상관분석 시행하여 다중공선성 확인

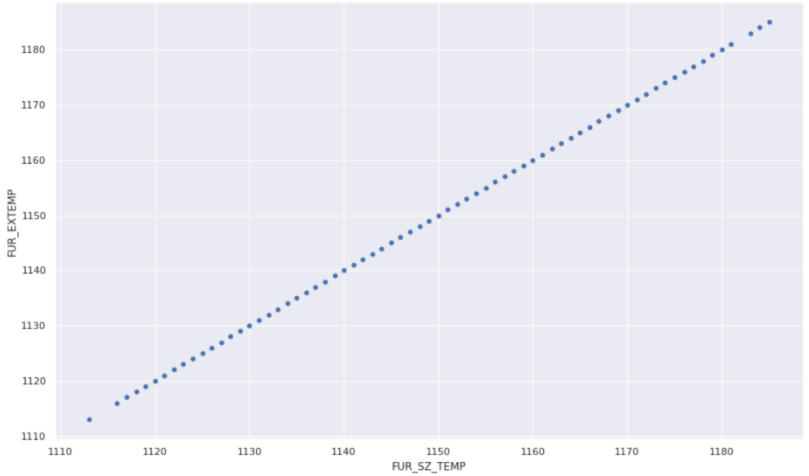
```
#이변수 분석
#=>변수 2개간의 관계 분석
import seaborn as sns

# 상관 관계 계수 시각화
sns.set(rc = {'figure.figsize':(15,9)})

mask = np.triu(np.ones_like(corr, dtype=np.bool))
sns.heatmap(corr, annot=True, cmap="Blues", mask=mask) # 숫자 넣어주기 : annot=True, 절반만 나오게 하기 : mask
```



1-2. RAW data확인 - 연속형 변수에 대한 상관분석 시행하여 다중공선성 확인

FUR_SZ_TEMP, FUR_EXTEMP 두 변수간의 상관관계가 1이어서 scatterplot을 이용하여 분포를 살펴봄

```
# 두 변수간의 상관관계가 1이어서 scatterplot을 이용하여 분포를 살펴봄
sns.scatterplot(data=df_raw, x="FUR_SZ_TEMP",y="FUR_EXTEMP")
```




```
# 결측치 확인
print(df_raw["FUR_SZ_TEMP"].isnull().sum())
print(df_raw["FUR_EXTEMP"].isnull().sum())

# => 둘다 결측치가 0 이므로 FUR_SZ_TEMP 변수를 삭제 하기로 함
```

```
0
0
```

1-2. RAW data확인 - 연속형 변수에 대한 상관분석 시행하여 다중공선성 확인

FUR_SZ_TEMP, FUR_EXTEMP 두 변수간의 상관관계가 1이어서 scatterplot을 이용하여 분포를 살펴봄

* 다중 공선성(multicollinearity)
두개의 독립변수간에 상관관계가 매우 커서, 즉 한 변수를 알면 자동적으로 다른 변수를 알 수 있을 때에는 다변량 분석이라도 이 두개의 독립변수가 종속변수에 미치는 영향을 따로 분리해서 산출할 수는 없다.

*다중공선성 확인 지표 : 분산 팽창계수 (VIF)
* VIF = 1/(1-R^2) 의 값이 5이상이면 다중공선성이 존재한다고 할 수 있음
*결정계수 (R^2) = (상관계수)^2

*다중 공선성이 있는 변수(=상관성이 높은 변수)들 처리 방법
1. 그 변수를 모델에서 제외한다. => 두 변수 중 결측치가 더 많은 변수 제외하기
2.“and/or" 조합을 이용한다.
3. 척도(scale)를 만든다. => 두 변수를 종합하는 새로운 변수를 만듦

1-2. RAW data확인 - 연속형 변수에 대한 상관분석 시행하여 다중공선성 확인

FUR_SZ_TEMP, FUR_EXTEMP 두 변수간의 상관관계가 1이어서 scatterplot을 이용하여 분포를 살펴봄

```
# 결측치 확인
print(df_raw["FUR_SZ_TEMP"].isnull().sum())
print(df_raw["FUR_EXTEMP"].isnull().sum())

# => 둘다 결측치가 0 이므로 FUR_SZ_TEMP 변수를 삭제 하기러 함
```

```
0
0
```

```
# 다중공선성 존재하는 두 변수 중 FUR_SZ_TEMP 삭제
df_raw.drop(['FUR_SZ_TEMP'], axis=1, inplace=True)
```

1-2. RAW data확인 - 범주형 변수에 대한 카테고리별 양품, 불량 분포 확인

```
# 범주형 변수에 대한 카테고리별 양품, 불량 개수 확인

def fun_print_crosstab(data, var):
    print(pd.crosstab(index = data["SCALE"], columns = data[var]))
    print()
    print(pd.crosstab(index = data["SCALE"],columns = data[var], normalize ="columns").round(3))
```

```
# 강종
fun_print_crosstab(df_raw, "STEEL_KIND")

#=> C0 강종은 다른 강종에 비해 상대적으로 높은 불량률을 보이고 있음 ; 조사할 필요 있음
```

```
STEEL_KIND   C0  C1  C3  T0  T1  T3  T5  T7  T8
SCALE
0           291   0   6  14  16   2  41  29  90
1           212   1   1   2   2   0   2   6   5

STEEL_KIND     C0   C1     C3     T0     T1   T3     T5     T7     T8
SCALE
0           0.579  0.0  0.857  0.875  0.889  1.0  0.953  0.829  0.947
1           0.421  1.0  0.143  0.125  0.111  0.0  0.047  0.171  0.053
```

C0 강종은 다른 강종에 비해 상대적으로 높은 불량률을 보이고 있음

1-2. RAW data확인 - 범주형 변수에 대한 카테고리별 양품, 불량 분포 확인

```
1 # 가열로 호기
2 fun_print_crosstab(df_raw, "FUR_NO")
```

```
FUR_NO  1호기  2호기  3호기
SCALE
0       167   167   155
1        73    70    88

FUR_NO    1호기    2호기    3호기
SCALE
0       0.696  0.705  0.638
1       0.304  0.295  0.362
```

```
1 # 작업조
2 fun_print_crosstab(df_raw, "WORK_GR")
```

```
WORK_GR   1조   2조   3조   4조
SCALE
0        122  120  118  129
1         67   45   54   65

WORK_GR      1조      2조      3조      4조
SCALE
0        0.646  0.727  0.686  0.665
1        0.354  0.273  0.314  0.335
```

[첨부]데이터 전처리

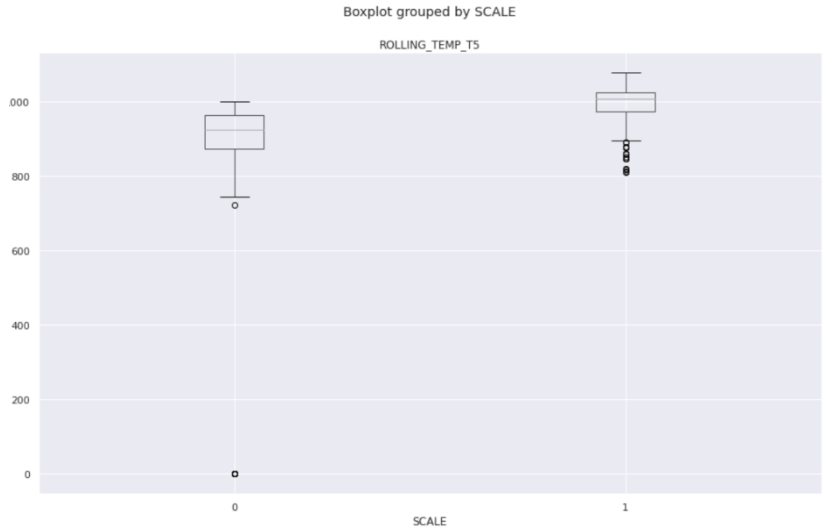
2. 결측값처리

```
# df_raw 데이터의 결측값 확인

df_raw.isnull().sum()
#=> 결측값 없음
```

```
PLATE_NO            0
ROLLING_DATE        0
SCALE               0
SPEC                0
STEEL_KIND          0
PT_THK              0
PT_WDTH             0
PT_LTH              0
PT_WGT              0
FUR_NO              0
FUR_NO_ROW          0
FUR_HZ_TEMP         0
FUR_HZ_TIME         0
FUR_SZ_TIME         0
FUR_TIME            0
FUR_EXTEMP          0
ROLLING_TEMP_T5     0
HSB                 0
ROLLING_DESCALING   0
WORK_GR             0
dtype: int64
```

3. 이상값처리

```
# ROLLING_TEMP_T5 에 대해 boxplot 그려봄
df_raw.boxplot(column = "ROLLING_TEMP_T5", by = "SCALE")

# => 0 이 존재함 :  ROLLING_TEMP_T5 은 압연온도이기에 0 이 존재할 수 없음 : 완벽한 이상치임 => 제거
```

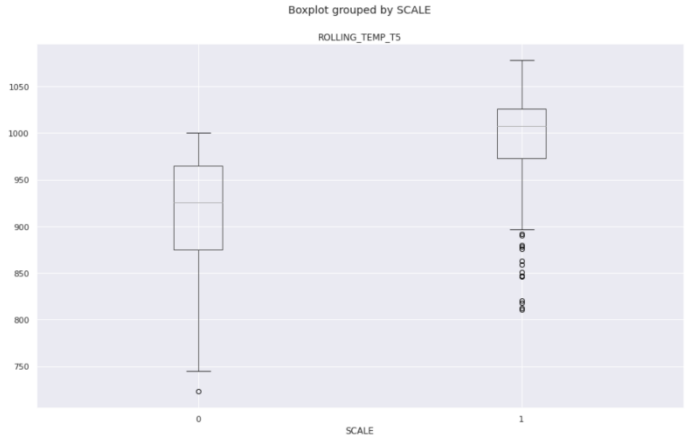


[첨부]데이터 전처리

3. 이상값처리

```
# ROLLING_TEMP_T5 가 600 이상인 데이터만 남김

df_raw = df_raw[df_raw.ROLLING_TEMP_T5>600]
```

```
# 다시 ROLLING_TEMP_T5 컬럼의 분포에 대해 boxplot 찍어봄
df_raw.boxplot(column = "ROLLING_TEMP_T5", by = "SCALE")
```



4. Feature Engineering – 불필요한 변수 삭제 및 독립, 종속 변수 나누기

```
# 일단 불필요한 변수 ROLLING_DATE, PLATE_NO, SPEC 삭제
df_raw.drop(['ROLLING_DATE'], axis=1, inplace=True)
df_raw.drop(['PLATE_NO'], axis=1, inplace=True)
df_raw.drop(['SPEC'], axis=1, inplace=True)
```

```
# 독립, 종속 변수 나누기

df_y = df_raw['SCALE']
df_x = df_raw.drop(["SCALE"], axis = 1, inplace = False)
```

4. Feature Engineering – get_dummies 함수를 이용하여 범주형 변수 가변수화

```
# 머신러닝을 할 때 기계가 이해할 수 있도록 모든 데이터를 수치로 변환해주는 전처리 작업이 필수적이다
# 수치형 데이터로만 변환을 하게 되면 서로 간의 관계성이 생기게 되므로 가변수화 작업이 필요!!!
# => pandas의  get_dummies함수 이용
df_x_dummy = pd.get_dummies(df_x)
df_x_dummy.head()

# STEEL_KIND, FUR_NO, WORK_GR 이 object 변수이었는데, get_dummies함수를 통해서 0, 1 값으로 변경된 것을 볼수있음
# 참고 : https://devuna.tistory.com/67
```

TEMP	ROLLING_TEMP_T5	HSB	ROLLING_DESCALING	STEEL_KIND_C0	STEEL_KIND_C1	STEEL_KIND_C3	STEEL_KIND_T0	STEEL_KIND_T1	STEEL_KIND_T3	STEEL_KIND_T5	STEEL_KIND_T7	STEEL_KIND_T8	FUR_NO_1	FUR_NO_2	FUR_NO_3	WORK_GR_1	WORK_GR_2	W
1133	934	1	8	0	0	0	0	1	0	0	0	0	1	0	0	0	1	
1135	937	1	8	0	0	0	0	1	0	0	0	0	1	0	0	0	1	
1121	889	1	8	0	0	0	0	0	0	0	0	1	0	1	0	0	0	
1127	885	1	8	0	0	0	0	0	0	0	0	1	0	1	0	0	0	
1128	873	1	8	0	0	0	0	0	0	0	0	1	0	0	1	1	0	

4. Feature Engineering – train set : test set = 7 : 3 분할

```
# train, test set => 0.7:0.3 으로 분할

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(df_x_dummy, df_y, test_size = 0.3, random_state = 1234)

print("train data x size : {}".format(x_train.shape))
print("train data y size : {}".format(y_train.shape))
print("test data x size : {}".format(x_test.shape))
print("test data y size : {}".format(y_test.shape))
```

```
train data x size : (499, 29)
train data y size : (499,)
test data x size : (215, 29)
test data y size : (215,)
```

1. 분류모델(의사결정나무)

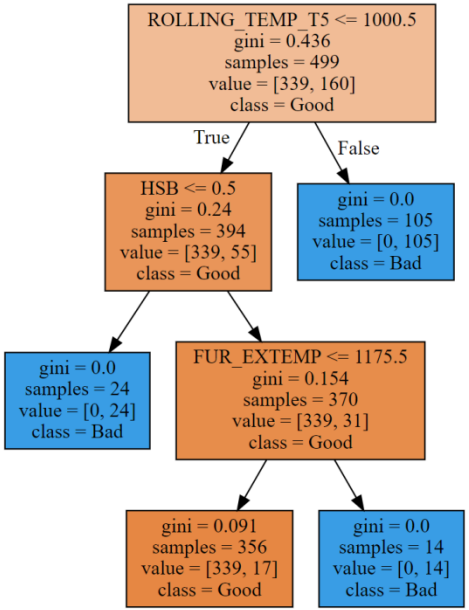
```
#최종모델
tree_final = DTC(random_state = 1234, max_depth = 3,min_samples_split=20, min_samples_leaf=14)
tree_final.fit(x_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=3, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=14, min_samples_split=20,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=1234, splitter='best')
```

```
#변수명 저장
v_feature_name = x_train.columns
```

```
# tree_final.dot으로 결과 저장 (목표변수 0: good , 1 : bad)
export_graphviz(tree_final, out_file = "tree_final.dot", class_names = ["Good", "Bad"],feature_names = v_feature_name, impurity = True, filled= True)

# graphviz 를 이용해 트리 모델 시각화
with open("tree_final.dot") as f:
    dot_graph = f.read()

display(graphviz.Source(dot_graph))
```

1. 분류모델(의사결정나무)



ROLLING_TEMP_T5(압연온도)가 1000.5 도 보다 높은 경우, HBS를 적용하지 않은 경우, FUR_EXTEMP(추출온도) 가 1175.5 보다 높은 경우 불량 발생

1. 분류모델(의사결정나무)

```
# 최종 모델 평가

y_pred = tree_final.predict(x_test)
print("Train Accuracy: {0:.3f}₩n".format(tree_final.score(x_train,y_train)), end ="")
print("Test Accuracy: {0:.3f}₩n".format(tree_final.score(x_test,y_test)))
```

```
Train Accuracy: 0.966
Test Accuracy: 0.953
```

```
print("Confusion matrix56: ₩n{}".format(confusion_matrix(y_test,y_pred)))

# => 정분류율 95.35%
```

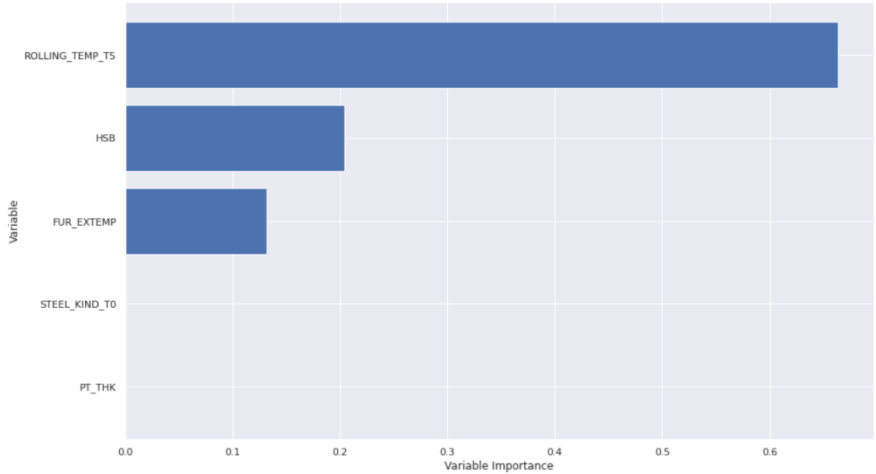
```
Confusion matrix56:
[[144   0]
 [ 10  61]]
```

```
# 결론도출
#tree.feature_importances_로 설명변수 중요도 확인 및 테이블로 저장

df_importance = pd.DataFrame()
df_importance["Feature"] = v_feature_name
df_importance["Importance"] = tree_final.feature_importances_

df_importance.sort_values("Importance", ascending = False, inplace = True)
df_importance5 = df_importance[:5]
df_importance5.round(4)
```

	Feature	Importance
10	ROLLING_TEMP_T5	0.6634
11	HSB	0.2045
9	FUR_EXTEMP	0.1320
0	PT_THK	0.0000
16	STEEL_KIND_T0	0.0000

1. 분류모델(의사결정나무)

```
#설명변수 중요도 그래프
#중요도가 높은 변수를 상위에 그림

df_importance5.sort_values("Importance", ascending = True, inplace = True)
coordinates = range(len(df_importance5))
plt.barh(y= coordinates, width =df_importance5["Importance"])
plt.yticks(coordinates, df_importance5["Feature"])
plt.xlabel("Variable Importance")
plt.ylabel("Variable")
```



[첨부]모델링

1. 분류모델(랜덤포레스트)

```
#결론 도출

#최종 모델
rf_final = RandomForestClassifier(min_samples_leaf = 5,min_samples_split=30, max_depth = 12, n_estimators = 58, random_state=1234)
rf_final.fit(x_train, y_train)

#평가
y_pred = rf_final.predict(x_test)

#train 데이터 셋 정확도
print("Accuracy on training set:{:.3f}".format(rf_final.score(x_train, y_train)))
#train 데이터 셋 정확도
print("Accuracy:on test set:{:.3f}₩n".format(rf_final.score(x_test, y_test)))
```

```
Accuracy on training set:0.956
Accuracy:on test set:0.907
```

```
#confusion matrix
print("Confusion matrix:₩n{}".format(confusion_matrix(y_test, y_pred)))

#=> 정분류율 90.7%
```

```
Confusion matrix:
[[144   0]
 [ 20  51]]
```

1. 분류모델(랜덤포레스트)

```
# 변수명 지정
v_feature_name = x_train.columns

#tree.feature_importances_로 설명변수 중요도 확인 및 테이블로 저장
df_importance = pd.DataFrame()
df_importance["Feature"]=v_feature_name
df_importance["Importance"]=rf_final.feature_importances_

#df_feature_importance의 테이블을 중요도 순으로 정렬
df_importance.sort_values('Importance', ascending = False, inplace = True)
df_importance10 = df_importance[:10]
df_importance10.round(3)
```

	Feature	Importance
10	ROLLING_TEMP_T5	0.401
9	FUR_EXTEMP	0.101
11	HSB	0.086
1	PT_WDTH	0.070
0	PT_THK	0.064
2	PT_LTH	0.047
13	STEEL_KIND_C0	0.044
5	FUR_HZ_TEMP	0.042
12	ROLLING_DESCALING	0.037
7	FUR_SZ_TIME	0.028

1. 분류모델(랜덤포레스트)

```
#설명변수 중요도 그래프 그리기
#중요도가 높은 변수를 상위에 그림
df_importance10.sort_values("Importance", ascending=True, inplace=True)
coordinates = range(len(df_importance10))
plt.barh(y = coordinates, width = df_importance10["Importance"])
plt.yticks(coordinates, df_importance10["Feature"])
plt.xlabel("importance")
plt.ylabel("columns")
```