종합실습 2

A2 김승희

분석 계획

01. Data Processing
- 데이터 특성 파악,
- 이상치 및 결측치 처리

02. EDA
- 범주형 설명변수 카이제곱 검정
- 연속형 설명변수 분포 확인

03. Modeling
- 로지스틱 회귀분석, 의사결정나무,
  랜덤포레스트, 그라디언트부스팅, MLP

04. Conclusion
- 모델 평가 및 의견

Data Processing

- **사전 조사에 의하여 목표변수와 상관없다고 여겨지는 열 삭제**

01. **Plate_no : plate 번호**
    - 제품번호는 제조과정에 전혀 관계가 없음
02. **rolling_date : 열연작업시각**
    - 자동화 공정이므로 작업시각 자체는 관계가 없다고 판단
03. **spec_long : 제품 규격**
    - 제품 규격은 철강의 제조과정과 관계가 없다고 판단
04. **spec_country ; 제품 규격 기준국**
    - 위와 마찬가지로 규격 기준 잦는 제조 과정과 관계가 없다고 판단
11. **fur_input_row : 가입로 장입열**
    - 재료의 균질화에 기여하는 요소가 아니기 때문에 제거해도 된다고 판단
21. **work_group : 작업조**
    - 제조공정 자동화 시스템이므로 작업조 자체는 불량률과 관계가 없다고 판단

Data Processing

• 결측치 확인

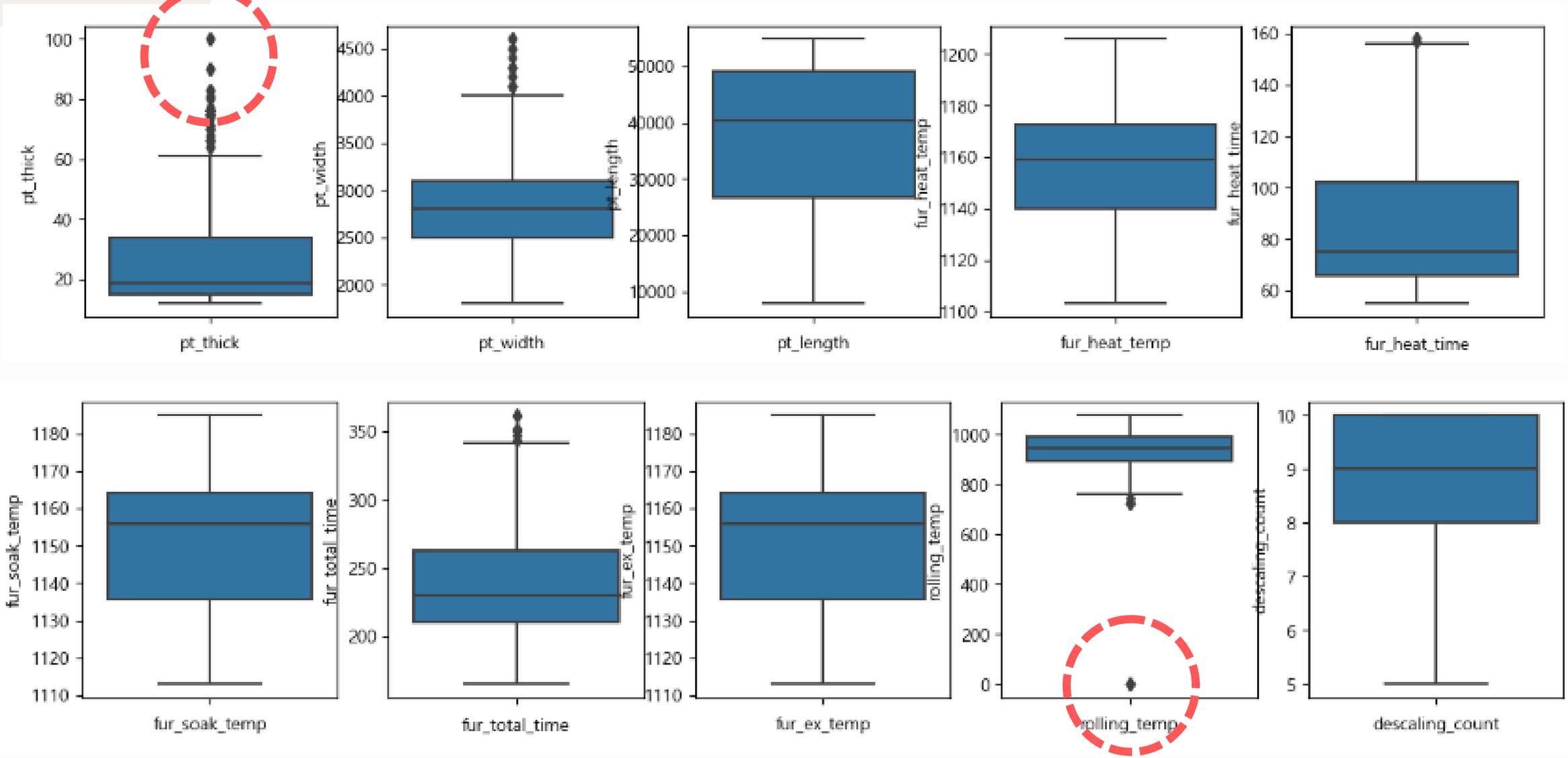
= 결측치 없음을 확인

```
1  #결측치 확인
2  df_raw.isnull().sum()
```

```
scale              0
steel_kind         0
pt_thick           0
pt_width           0
pt_length          0
hsb                0
fur_no             0
fur_heat_temp      0
fur_heat_time      0
fur_soak_temp      0
fur_soak_time      0
fur_total_time     0
fur_ex_temp        0
rolling_method     0
rolling_temp       0
descaling_count    0
dtype: int64
```

Data Processing

• 수치형 설명변수 이상치 확인
--> pt_thick, rolling_temp에서 이상치 확인



Data Processing

. 파생 변수 생성

```
#파생 변수 생성
# 크기 = 후판 지시두께*(지시폭+지시길이)
df_raw['size'] = df_raw['pt_thick'] *(df_raw['pt_width'] + df_raw['pt_length'])
df_raw = df_raw.drop(['pt_thick', 'pt_width', 'pt_length'], axis=1)
```

재료 안에 균질을 하려면 재료의 중심부까지 열이 전달이 되어야 하는데
부피가 너무 크면 열이 전달이 잘 안될 것이라고 생각하였다.

따라서 지시 두께, 지시폭, 지시길이 변수를 이용하여 새롭게 크기라는 변수를 생성하였다.

탐색적 분석

• 범주형 설명 변수에 대한 검정

```
# 설명변수 중 object형이 변수 확인
print(df_raw['steel_kind'].unique())
print(df_raw['hsb'].unique())
print(df_raw['fur_no'].unique())
print(df_raw['rolling_method'].unique())
```

```
['T' 'C']
['적용' '미적용']
['1호기' '2호기' '3호기']
['TMCP(온도제어)' 'CR(제어압연)']
```

• 총 4개의 범주형 설명 변수가 있다.

• 각각의 설명변수들이 scale에 영향이 있는지,
범주별로 차이가 있는지에 대해 검정하고자 한다

=> 카이제곱검정 시행

탐색적 분석

• 범주형 설명 변수에 대한 검정

```
1 chisquare(df_raw['fur_no'], 'fur_no')
```

```
Contingency Table:
fur_no  1호기  2호기  3호기  All
scale
0        231   231   219   681
1        100    92   116   308
All      331   323   335   989

카이제곱 통계량: 3.10
P-값: 0.80
 scale에 영향을 미치지 않습니다.
```

fur_no : 가열로 호기는
호기 별로 scale에 차이가 없다
=> 제거하기로 결정

```
1 chisquare(df_raw['rolling_method'], 'rolling_method')
```

```
Contingency Table:
rolling_method  CR(제어압연)  TMCP(온도제어)  All
scale
0                    539          142  681
1                    295           13  308
All                  834          155  989

카이제곱 통계량: 44.38
P-값: 0.00
 scale에 영향을 미칩니다.
```

```
1 chisquare(df_raw['hsb'],'hsb')
```

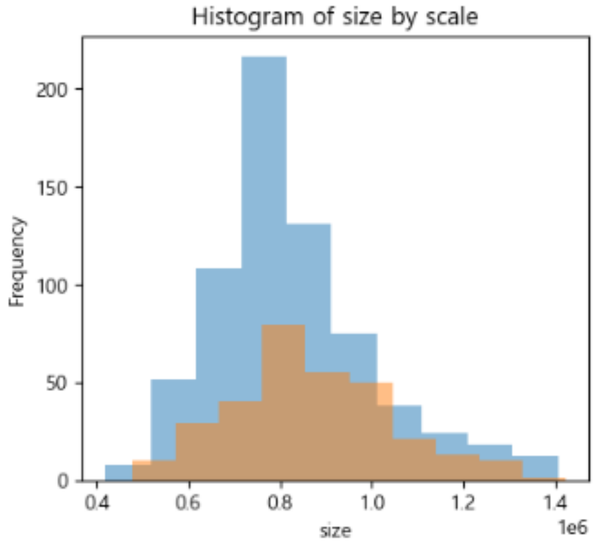
```
Contingency Table:
hsb     미적용   적용  All
scale
0         0   681  681
1        47   261  308
All      47   942  989

카이제곱 통계량: 109.10
P-값: 0.00
 scale에 영향을 미칩니다.
```

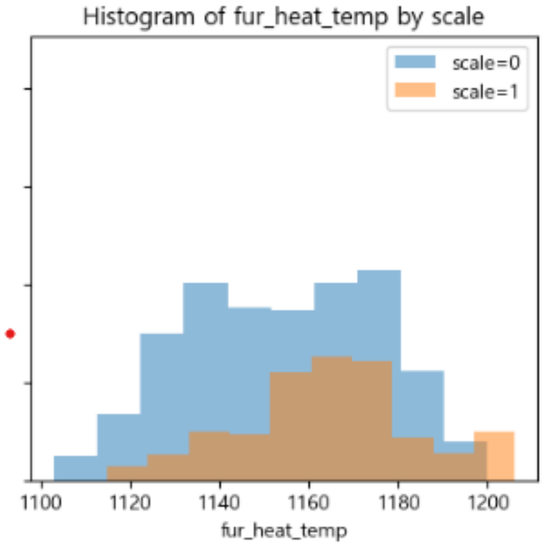
```
1 chisquare(df_raw['steel_kind'], 'steel_kind')
```

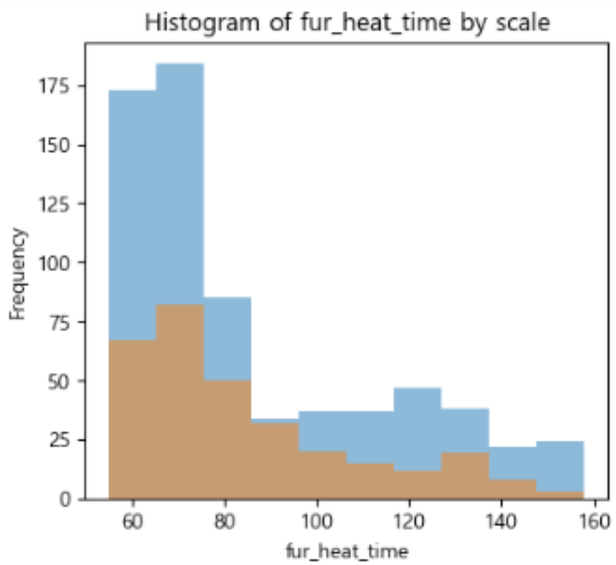
```
Contingency Table:
steel_kind    C    T  All
scale
0           463  218  681
1           288   20  308
All         751  238  989

카이제곱 통계량: 75.58
P-값: 0.00
 scale에 영향을 미칩니다.
```

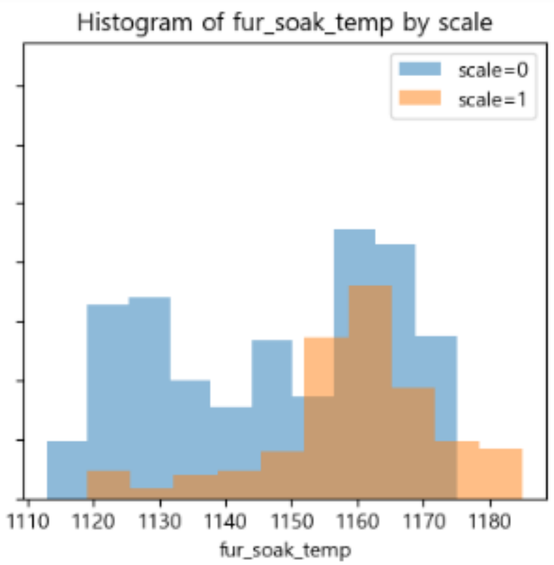
탐색적 분석

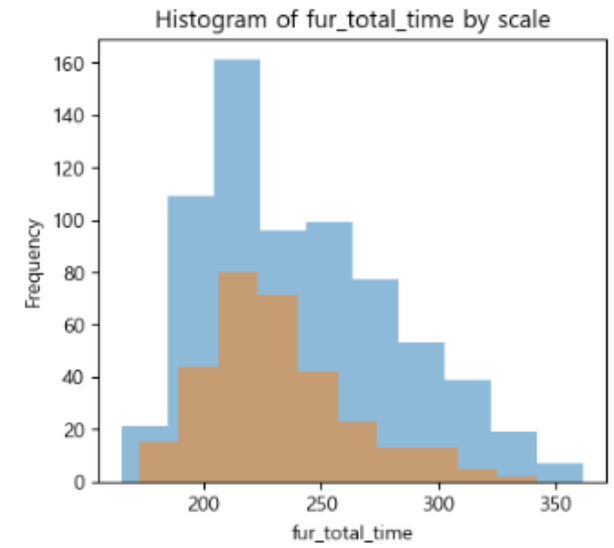
• 연속형 설명 변수에 대한 분포 확인

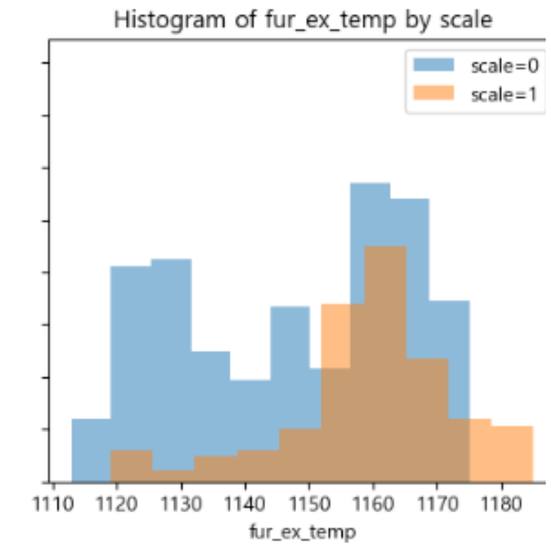


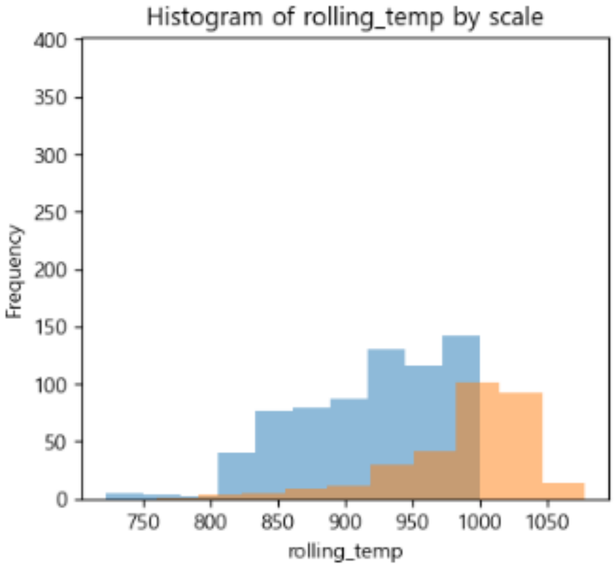


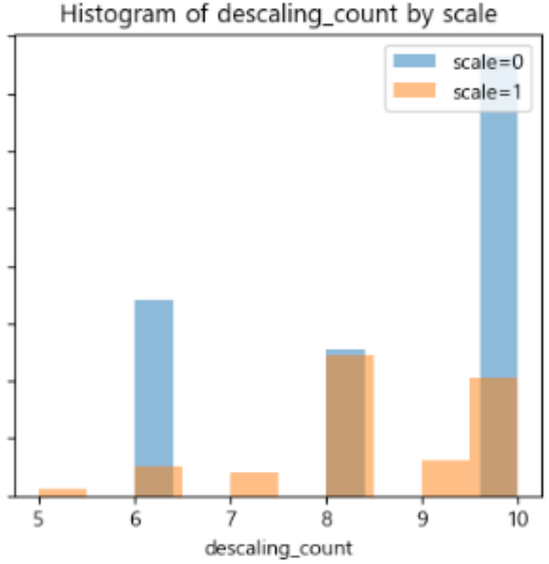












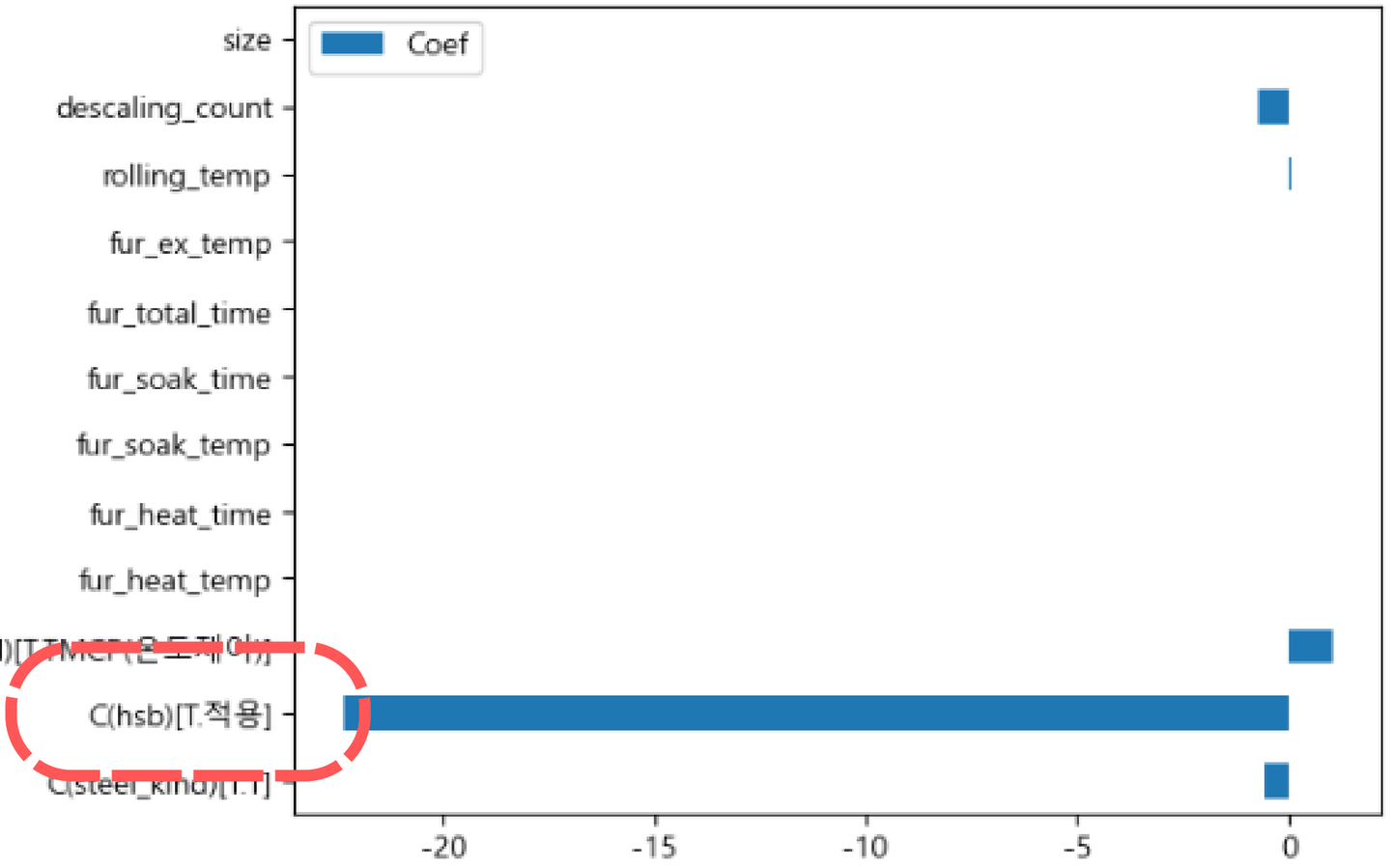
Modeling

- 로지스틱 회귀
  수치형 설명변수 scale한 뒤 로지스틱 회귀 진행

과적합 X

```
Trian 예측/분류 결과
Accuracy: 0.863

Test 예측/분류 결과
Accuracy: 0.862
```

```
Confusion Matrix:
[[183  26]
 [ 15  73]]
```



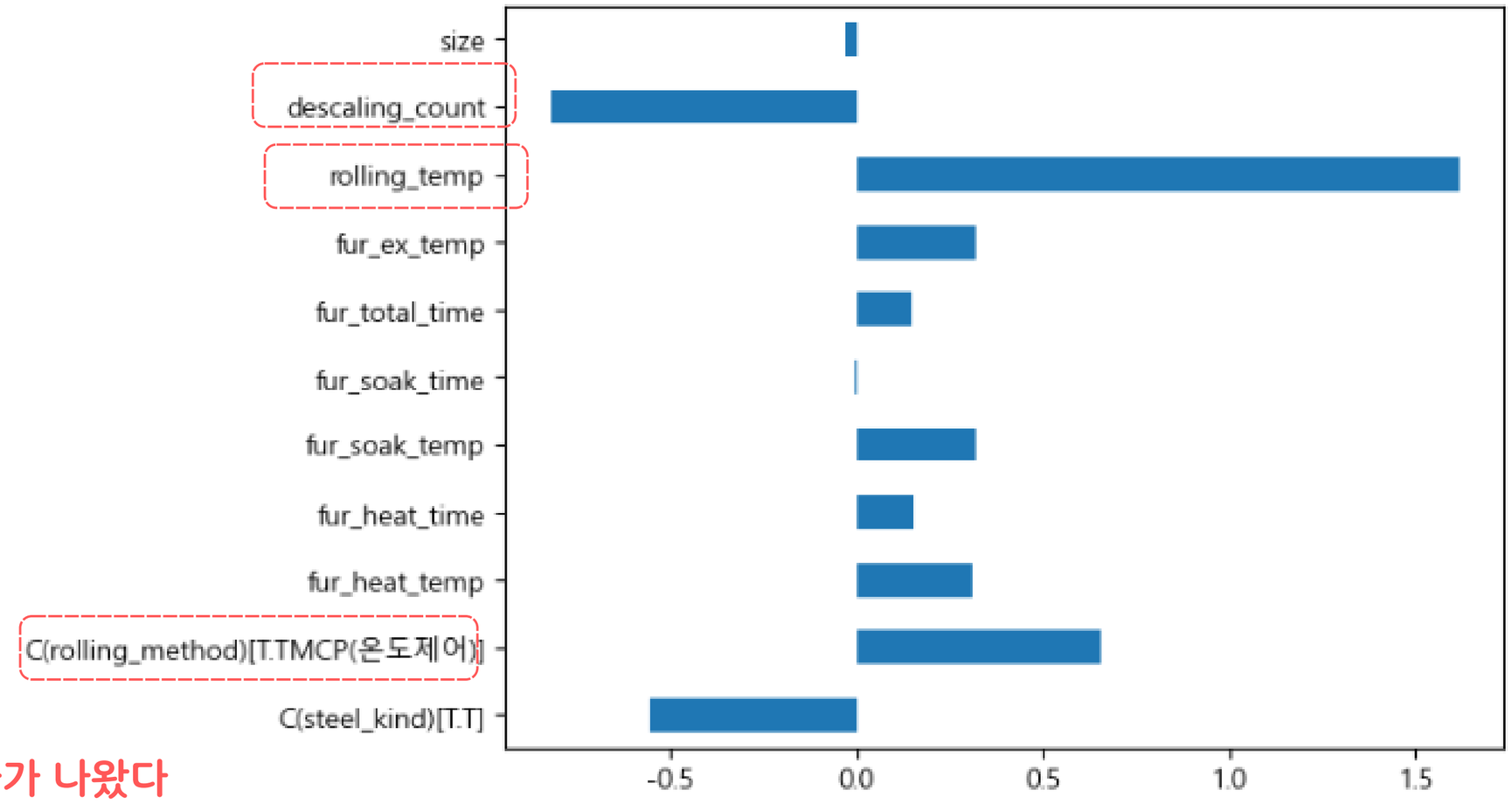
- hsb T.적용이 우선적으로 고려되어야 함
- hsb 설명변수에서 T.적용이 너무 중요도가 높게 나와 다른 설명변수들의 영향을 정확히 알 수 없으므로 이 변수를 제거 한 뒤 다시 로지스티 회귀 진행

Modeling

- 로지스틱 회귀
  hsb 변수 제거한 뒤 로지스틱 회귀 진행

과적합 X

```
Trian 예측/분류 결과
Accuracy: 0.682

Test 예측/분류 결과
Accuracy: 0.862

Confusion Matrix:
[[183  26]
 [ 15  73]]
```



- rolling_temp : 압연온도
- descaling_count : 압연 횟수
- rolling_method : 압연 방법

순으로 영향력 있는 인자라는 결과가 나왔다

Modeling

• 의사결정나무

```
1  tree_uncustomized = DecisionTreeClassifier(random_state = 1234)
2  tree_uncustomized.fit(df_train_x, df_train_y)
3
4  print("Score on training set: {:.3f}".format(tree_uncustomized.score(df_train_x, df_train_y)))
5  print("Score on test set: {:.3f}".format(tree_uncustomized.score(df_test_x, df_test_y)))
```

```
Score on training set: 1.000
Score on test set: 0.993
```

RandomizedSearchCV를 통해
parameter 조절
=> train데이터에 과대적합 해결

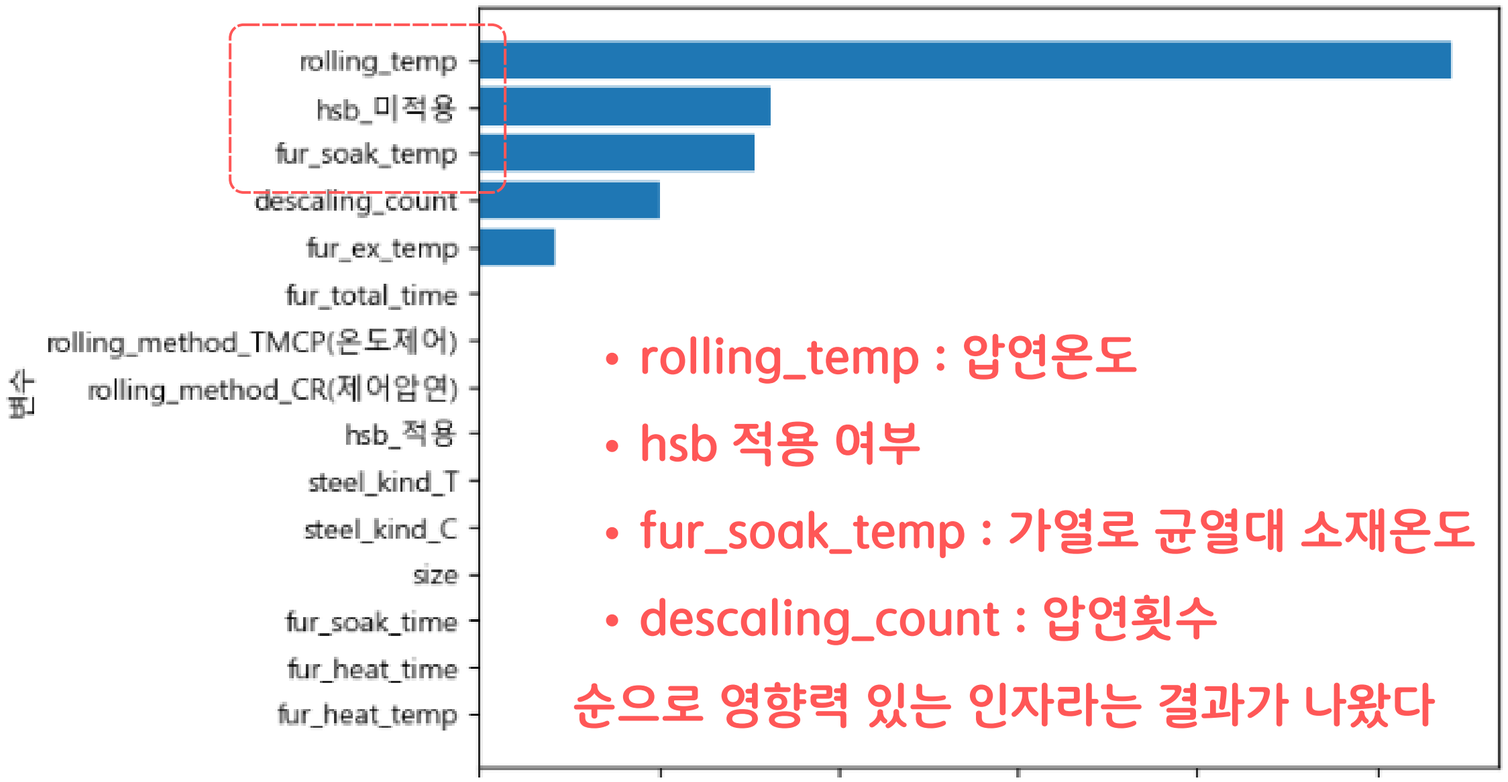
```
1  tree_final = DecisionTreeClassifier(min_samples_leaf=7,min_samples_split=19,max_depth=9, random_state=1234)
2  tree_final.fit(df_train_x, df_train_y)
3
4
5  print("Score on training set: {:.3f}".format(tree_final.score(df_train_x, df_train_y)))
6  print("Score on test set: {:.3f}".format(tree_final.score(df_test_x, df_test_y)))
```

```
Score on training set: 0.993
Score on test set: 0.993
```

Modeling

- 의사결정나무

	Feature	importance
6	rolling_temp	0.541
11	hsb_미적용	0.163
2	fur_soak_temp	0.153
7	descaling_count	0.100
5	fur_ex_temp	0.042
4	fur_total_time	0.001
0	fur_heat_temp	0.000
1	fur_heat_time	0.000
3	fur_soak_time	0.000
8	size	0.000
9	steel_kind_C	0.000
10	steel_kind_T	0.000
12	hsb_적용	0.000
13	rolling_method_CR(제어압연)	0.000
14	rolling_method_TMCP(온도제어)	0.000



- rolling_temp : 압연온도
- hsb 적용 여부
- fur_soak_temp : 가열로 균열대 소재온도
- descaling_count : 압연횟수

순으로 영향력 있는 인자라는 결과가 나왔다

Modeling

• 랜덤포레스트

```
rf_uncustomized = RandomForestClassifier(random_state=1234)
rf_uncustomized.fit(df_train_x, df_train_y)

print("Score on training set: {:.3f}".format(rf_uncustomized.score(df_train_x, df_train_y)))
print("Score on test set: {:.3f}".format(rf_uncustomized.score(df_test_x, df_test_y)))
```

```
Score on training set: 1.000
Score on test set: 0.963
```

RandomizedSearchCV를 통해
parameter 조절
=> train데이터에 과대적합 해결

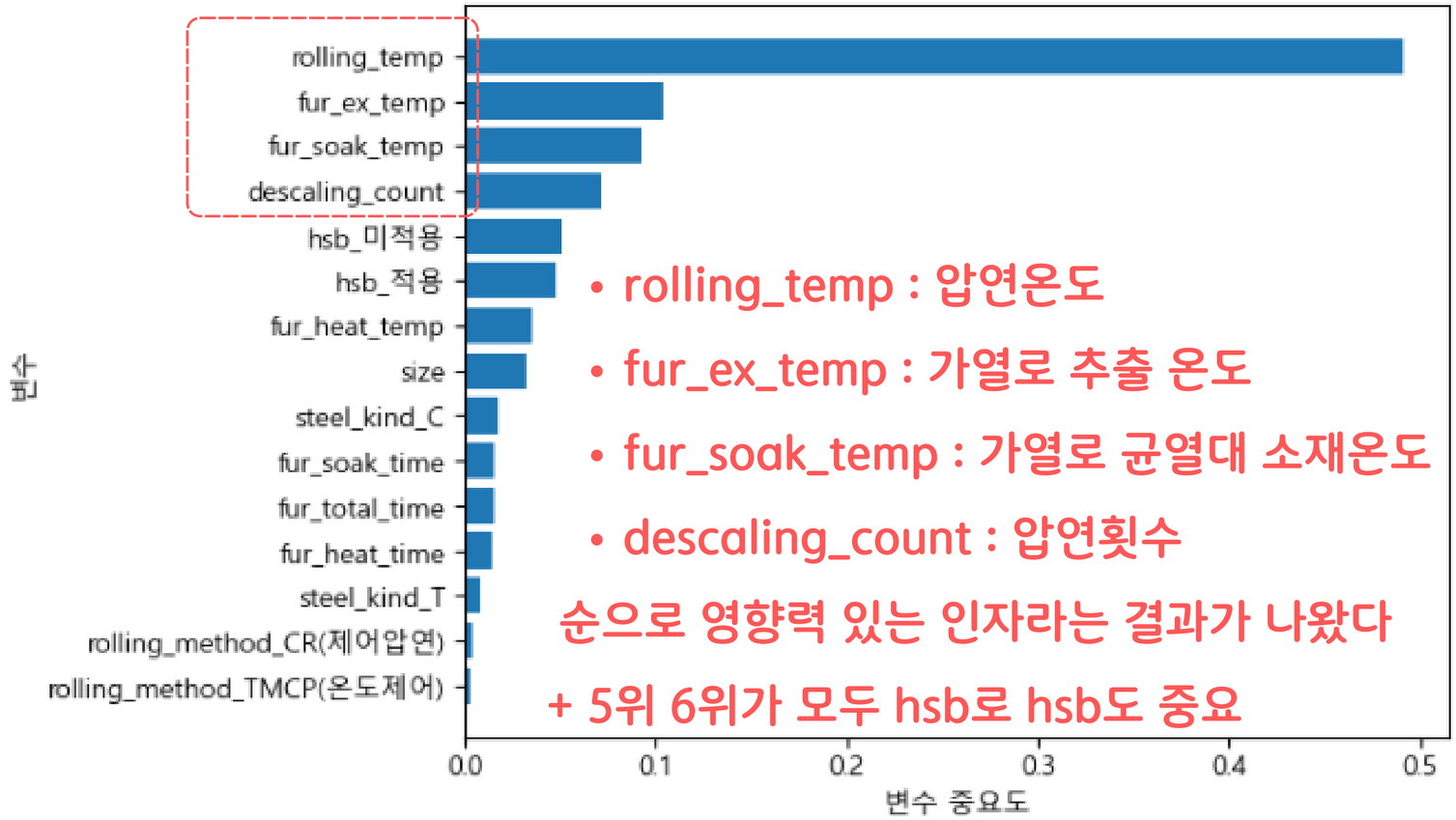
```
rf_final = RandomForestClassifier(min_samples_leaf = 11, min_samples_split = 9,
                                  max_depth = 15, random_state = 1234)
rf_final.fit(df_train_x, df_train_y)
y_pred = rf_final.predict(df_test_x)


print("Score on training set: {:.3f}".format(rf_final.score(df_train_x,df_train_y)))
print("Score on test set: {:.3f}".format(rf_final.score(df_test_x,df_test_y)))
```

```
Score on training set: 0.954
Score on test set: 0.939
```

Modeling

• 랜덤포레스트

	Feature	importance
6	rolling_temp	0.491
5	fur_ex_temp	0.103
2	fur_soak_temp	0.093
7	descaling_count	0.072
11	hsb_미적용	0.051
12	hsb_적용	0.047
0	fur_heat_temp	0.035
8	size	0.032
9	steel_kind_C	0.017
3	fur_soak_time	0.016
4	fur_total_time	0.015
1	fur_heat_time	0.014
10	steel_kind_T	0.009
13	rolling_method_CR(제어압연)	0.003
14	rolling_method_TMCP(온도제어)	0.002



- rolling_temp : 압연온도
- fur_ex_temp : 가열로 추출 온도
- fur_soak_temp : 가열로 균열대 소재온도
- descaling_count : 압연횟수

순으로 영향력 있는 인자라는 결과가 나왔다

+ 5위 6위가 모두 hsb로 hsb도 중요

Modeling

• 그라디언트부스팅

```python
# 모델 생성 : GradientboostingRegressor
gb_uncustomized = GradientBoostingClassifier(random_state=1234)
gb_uncustomized.fit(df_train_x, df_train_y)


print("Score on training set: {:.3f}".format(gb_uncustomized.score(df_train_x, df_train_y)))
print("Score on test set: {:.3f}".format(gb_uncustomized.score(df_test_x, df_test_y)))
```

```
Score on training set: 1.000
Score on test set: 0.993
```

RandomizedSearchCV를 통해
parameter 조절
=> train데이터에 과대적합 해결

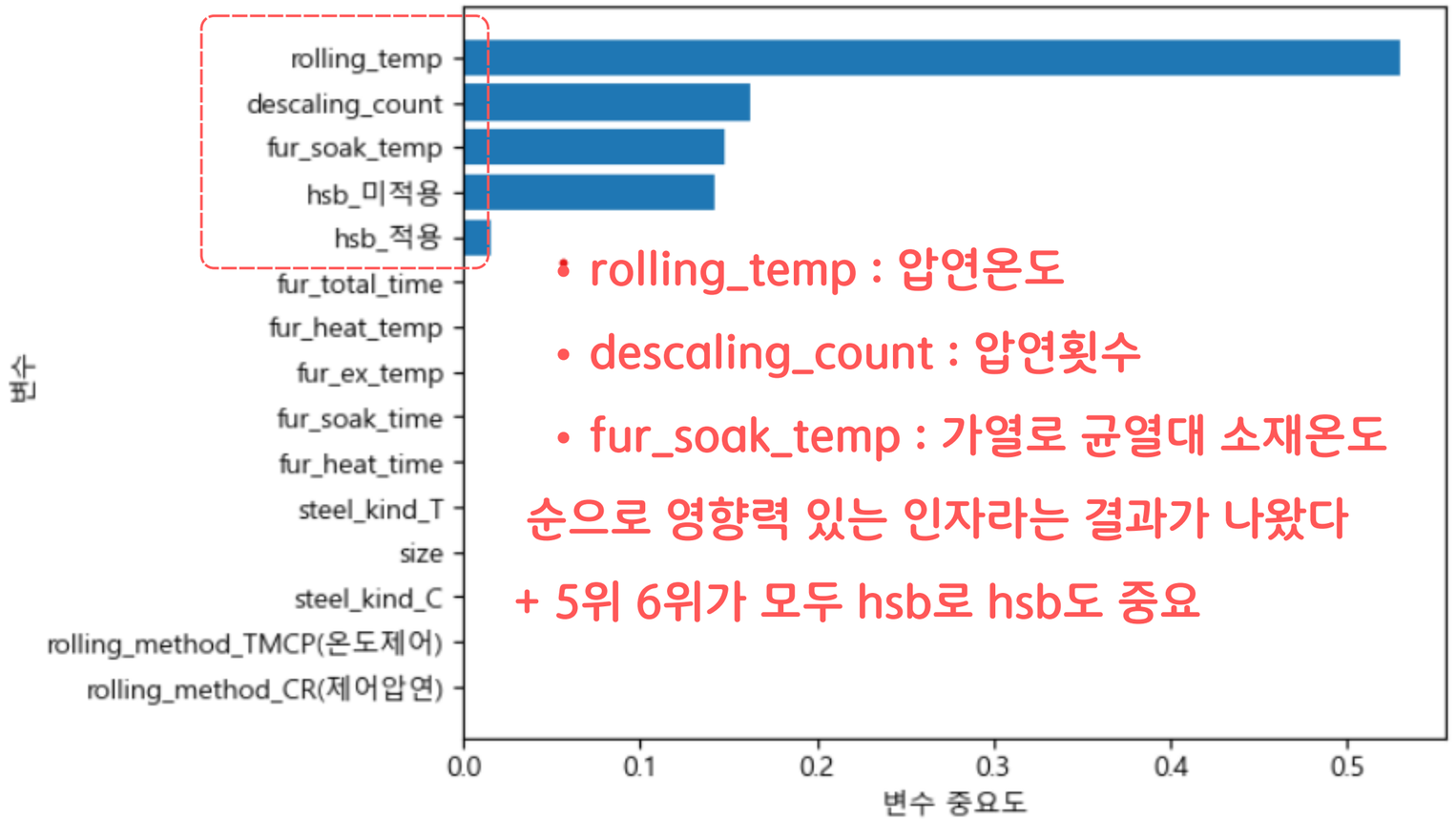
```python
# 그리디언트부스팅 최종 모델
gb_final = GradientBoostingClassifier(min_samples_leaf=4,max_depth=6, n_estimators=100,learning_rate=0.7, random_state=1234 )

gb_final.fit(df_train_x, df_train_y)
y_pred = gb_final.predict(df_test_x)


print("Accurcy on training set: {:.3f}".format(gb_final.score(df_train_x,df_train_y)))
print("Accuracy on test set: {:.3f}".format(gb_final.score(df_test_x,df_test_y)))
```

```
Accurcy on training set: 1.000
Accuracy on test set: 0.997
```

Modeling

• 그라디언트부스팅

	Feature	importance
6	rolling_temp	0.530
7	descaling_count	0.162
2	fur_soak_temp	0.147
11	hsb_미적용	0.142
12	hsb_적용	0.016
4	fur_total_time	0.001
0	fur_heat_temp	0.001
5	fur_ex_temp	0.001
3	fur_soak_time	0.000
1	fur_heat_time	0.000
10	steel_kind_T	0.000
8	size	0.000
9	steel_kind_C	0.000
13	rolling_method_CR(제어압연)	0.000
14	rolling_method_TMCP(온도제어)	0.000



- rolling_temp : 압연온도
- descaling_count : 압연횟수
- fur_soak_temp : 가열로 균열대 소재온도

순으로 영향력 있는 인자라는 결과가 나왔다

+ 5위 6위가 모두 hsb로 hsb도 중요

Modeling

• 인공신경망

```
v_feature_names = df_train_x.columns
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_raw_x)
df_scaled_train_x, df_scaled_test_x = train_test_split(df_scaled, test_size = 0.3, random_state = 1234)

print("Accuracy on training set: {:.3f}".format(nn_scaled.score(df_scaled_train_x,df_train_y)))
print("Accuracy on test set: {:.3f}".format(nn_scaled.score(df_scaled_test_x, df_test_y)))
```

```
Accuracy on training set: 0.575
Accuracy on test set: 0.559
```

RandomizedSearchCV를 통해
parameter 조절
=> train score는 유의미하게 상승했지만 test score는 변동 없음

```
# 최종 모델
mlp_final = MLPClassifier(hidden_layer_sizes=(137,), activation='logistic', solver='sgd',batch_size=100, random_state=1234)
mlp_final.fit(df_scaled_train_x, df_scaled_train_y)
y_pred = nn_final.predict(df_scaled_test_x)

print("Accuracy on training set: {:.3f}".format(nn_final.score(df_scaled_train_x, df_scaled_train_y)))
print("Accuracy on test set: {:.3f}".format(nn_scaled.score(df_scaled_test_x, df_scaled_test_y)))

```

```
Accuracy on training set: 0.727
Accuracy on test set: 0.559
```

Conclusion

일반적으로 불량품을 제조하는 공정과 같이 실제 Positive한 경우가 적은 경우에는 재현율을 더 중요하게 여기는 경우가 많다.

Decision Tree Confusion matrix :

```
[[186   3]
 [ 11 178]]
```

	precision	recall	f1-score	support
0	0.944	0.984	0.964	189
1	0.983	0.942	0.962	189
accuracy			0.963	378
macro avg	0.964	0.963	0.963	378
weighted avg	0.964	0.963	0.963	378

RandomForest Confusion matrix :

```
[[209   0]
 [ 18  70]]
```

	precision	recall	f1-score	support
0	0.921	1.000	0.959	209
1	1.000	0.795	0.886	88
accuracy			0.939	297
macro avg	0.960	0.898	0.922	297
weighted avg	0.944	0.939	0.937	297

GradientBoosting Confusion matrix :

```
[[209   0]
 [  1  87]]
```

	precision	recall	f1-score	support
0	0.995	1.000	0.998	209
1	1.000	0.989	0.994	88
accuracy			0.997	297
macro avg	0.998	0.994	0.996	297
weighted avg	0.997	0.997	0.997	297

MLP Confusion matrix :

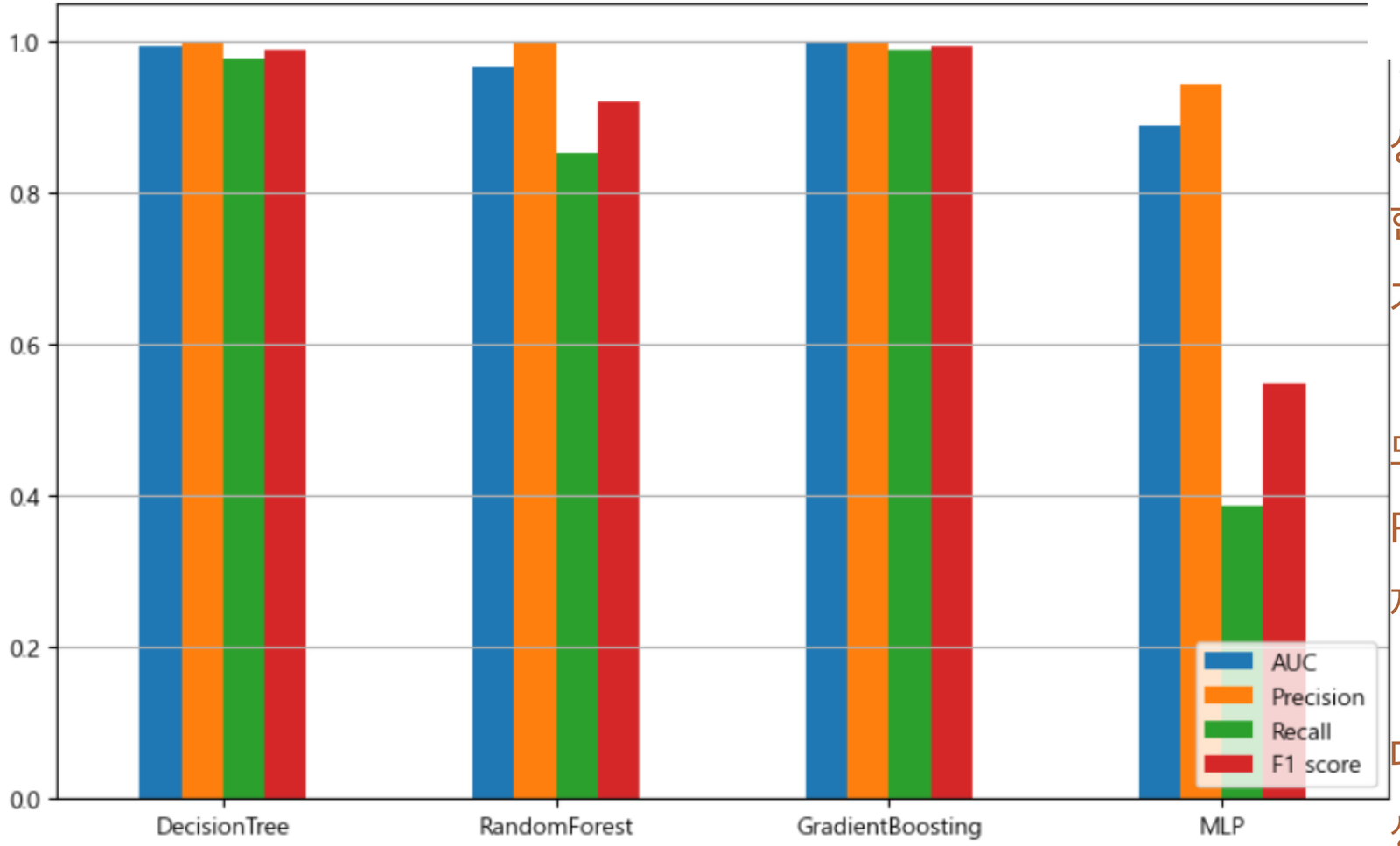
```
[[209   0]
 [ 72  16]]
```

	precision	recall	f1-score	support
0	0.744	1.000	0.853	209
1	1.000	0.182	0.308	88
accuracy			0.758	297
macro avg	0.872	0.591	0.580	297
weighted avg	0.820	0.758	0.691	297

GradientBoosting - RandomForest - DecisionTree-MLP 순으로 1에 대한 Recall값이 높다

=> 불량여부를 예측할 때 GradientBoosing이 가장 성능이 좋을 것이라 판단

Conclusion



	AUC	Precision	Recall	F1 score
DecisionTree	0.993	1.000	0.977	0.989
RandomForest	0.966	1.000	0.852	0.920
GradientBoosting	1.000	1.000	0.989	0.994
MLP	0.889	0.944	0.386	0.548

분류 모델의 성능이 좋을수록 ROC Curve는 왼쪽 상단 모서리로 갈수록 면적이 넓어져 AUC값이 증가한다. AUC값이 가장 큰 그라디언트부스팅이 가장 성능이 좋다고 할 수 있다.

또한 그라디언트부스팅을 사용할 경우에 Recall값도 제일 높으므로 불량 살펴볼 경우에 제일 정확도가 높다고 할 수 있다.

따라서 그라디언트부스팅의 결과를 우선적으로 생각하기로 판단하였다.

Conclusion

	Feature	importance
6	rolling_temp	0.530
7	descaling_count	0.162
2	fur_soak_temp	0.147
11	hsb_미적용	0.142
12	hsb_적용	0.016
4	fur_total_time	0.001
0	fur_heat_temp	0.001
5	fur_ex_temp	0.001
3	fur_soak_time	0.000
1	fur_heat_time	0.000
10	steel_kind_T	0.000
8	size	0.000
9	steel_kind_C	0.000
13	rolling_method_CR(제어압연)	0.000
14	rolling_method_TMCP(온도제어)	0.000

그라디언트 부스팅

	Feature	importance
6	rolling_temp	0.491
5	fur_ex_temp	0.103
2	fur_soak_temp	0.093
7	descaling_count	0.072
11	hsb_미적용	0.051
12	hsb_적용	0.047
0	fur_heat_temp	0.035
8	size	0.032
9	steel_kind_C	0.017
3	fur_soak_time	0.016
4	fur_total_time	0.015
1	fur_heat_time	0.014
10	steel_kind_T	0.009
13	rolling_method_CR(제어압연)	0.003
14	rolling_method_TMCP(온도제어)	0.002

랜덤포레스트

	Feature	importance
6	rolling_temp	0.541
11	hsb_미적용	0.163
2	fur_soak_temp	0.153
7	descaling_count	0.100
5	fur_ex_temp	0.042
4	fur_total_time	0.001
0	fur_heat_temp	0.000
1	fur_heat_time	0.000
3	fur_soak_time	0.000
8	size	0.000
9	steel_kind_C	0.000
10	steel_kind_T	0.000
12	hsb_적용	0.000
13	rolling_method_CR(제어압연)	0.000
14	rolling_method_TMCP(온도제어)	0.000

의사결정나무

모든 모델에서 rolling_temp : 압연온도, descaling_count : 압연횟수, fur_soak_temp : 가열로 균열 hsb적용 여부가 상단에 위치한다. 따라서 이 변수들이 scale불량 발생에 가장 큰 영향을 주는 인자들이다.

Conclusion

1. scale 불량 발생에 영향을 주는 인자 도출

   rolling_temp : 압연온도,
   descaling_count : 압연횟수
   fur_soak_temp : 가열로 균열
   hsb : hsb 적용 여부
   따라서 전체적으로 불량 발생에는 온도적인 요소가 굉장히 중요함이 알 수 있다.

2. 모델의 성능 개선 방향

   로지스틱 회귀분석의 결과에서 확인할 수 있듯이, hsb를 적용하지 않을 경우 무조건 불량이 발생하는데
   의사결정나무, 랜덤포레스트, 그라디언트 부스팅에서 hsb변수를 제거하고 모델링을 했을 경우
   수치형 변수들에 대한 더 정확한 영향력을 알 수 있을 것 같다.