후판 공정 scale 유발 영향인자 분석 및 개선안 도출

C반 권태준

Introduce project

분석 배경

최근들어 후판공정에서의 scale 불량이 급격히 증가함에 따라 이를 해결할 필요가 있다.

예상 목표

후판공정에서의 scale 발생에 영향을 미치는 주요 요인들을 파악하고 그에 따른 개선 방안을 제시해보고자 한다.

Introduce project

1. 탄소는 저렴하고 우수한 특성을 가져 강도와 인성이 뛰어나 많이 사용되고 있을 것임.
2. HSB는 불량률을 낮추기 위해 필수적으로 해야 하는 공정임.
3. 공정 온도가 높을수록 물질의 물성이 변형되거나 소성이 일어나 제품의 강도가 감소해 불량률이 높아질것임.
4. 공정의 경우 일반적으로 온도와 생산속도는 trade off관계를 가질것임.
5. 길이, 폭, 높이가 있으니 부피 변수를 만든다면 scale과의 연관성이 있을것임.

데이터 현황

```
# 결측치 확인
df_raw.isnull().sum(axis=0)
```

```
plate_no          0
rolling_date      0
scale             0
spec_long         0
spec_country      0
steel_kind        0
pt_thick          0
pt_width          0
pt_length         0
hsb               0
fur_no            0
fur_input_row     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
work_group        0
dtype: int64
```

결측치 없음

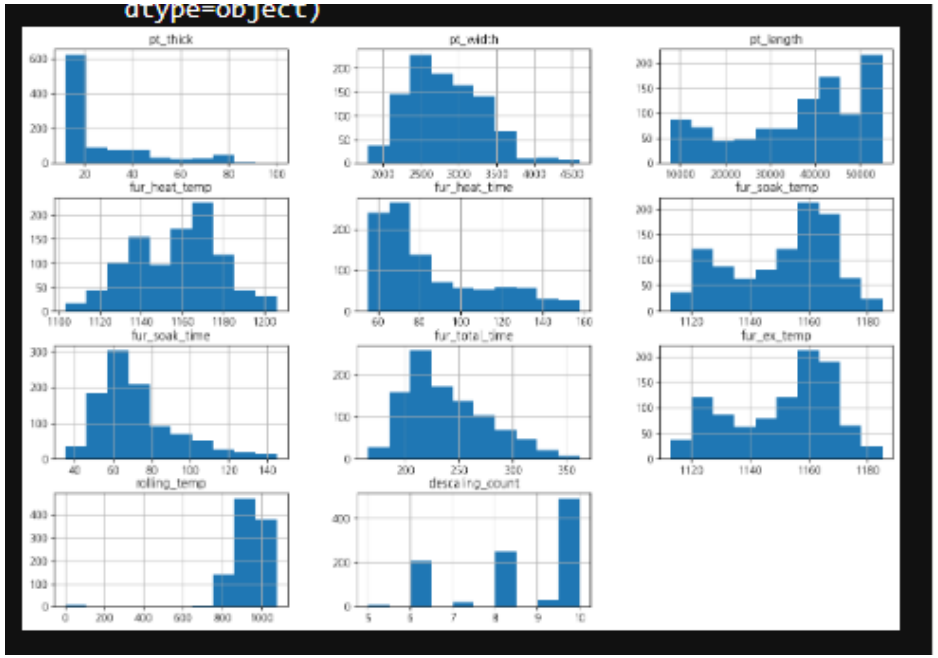
```
print('중복된 항목 수:',len(df_raw[df_raw.duplicated()]))
```

중복치 없음

```
df_raw.info()
```

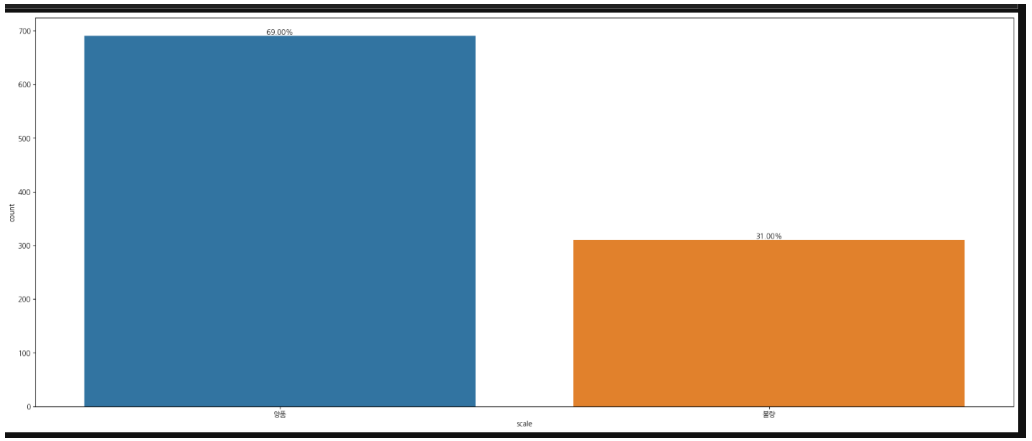
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   plate_no         1000 non-null   object
 1   rolling_date     1000 non-null   object
 2   scale            1000 non-null   object
 3   spec_long        1000 non-null   object
 4   spec_country     1000 non-null   object
 5   steel_kind       1000 non-null   object
 6   pt_thick         1000 non-null   int64
 7   pt_width         1000 non-null   int64
 8   pt_length        1000 non-null   int64
 9   hsb              1000 non-null   object
 10  fur_no           1000 non-null   object
 11  fur_input_row    1000 non-null   object
 12  fur_heat_temp    1000 non-null   int64
 13  fur_heat_time    1000 non-null   int64
 14  fur_soak_temp    1000 non-null   int64
 15  fur_soak_time    1000 non-null   int64
 16  fur_total_time   1000 non-null   int64
 17  fur_ex_temp      1000 non-null   int64
 18  rolling_method   1000 non-null   object
 19  rolling_temp     1000 non-null   int64
 20  descaling_count  1000 non-null   int64
 21  work_group       1000 non-null   object
dtypes: int64(11), object(11)
memory usage: 172.0+ KB
```

데이터 type 확인

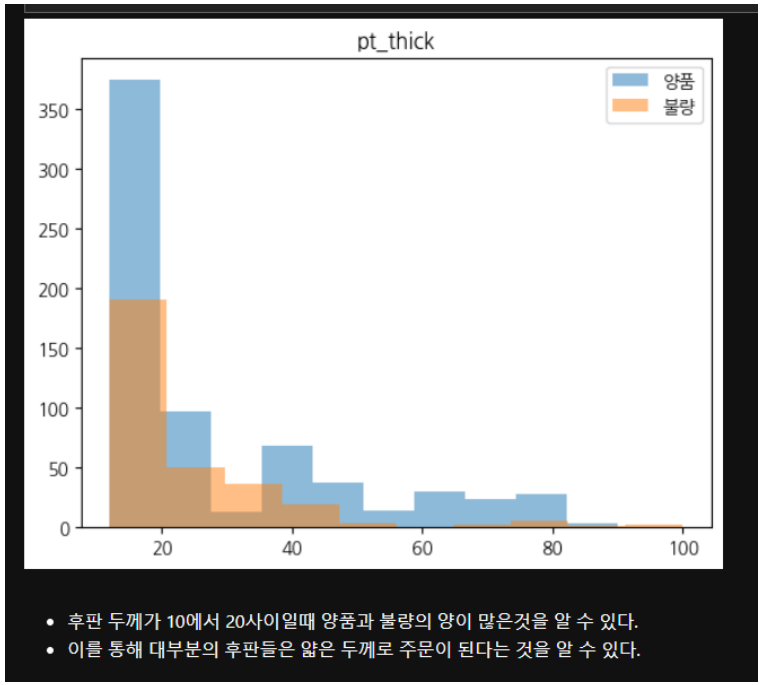


연속형 변수 분포 확인

Rolling_temp에서 0 부분에 데이터가 있음을 확인 / 이상치로 판단하여 제거

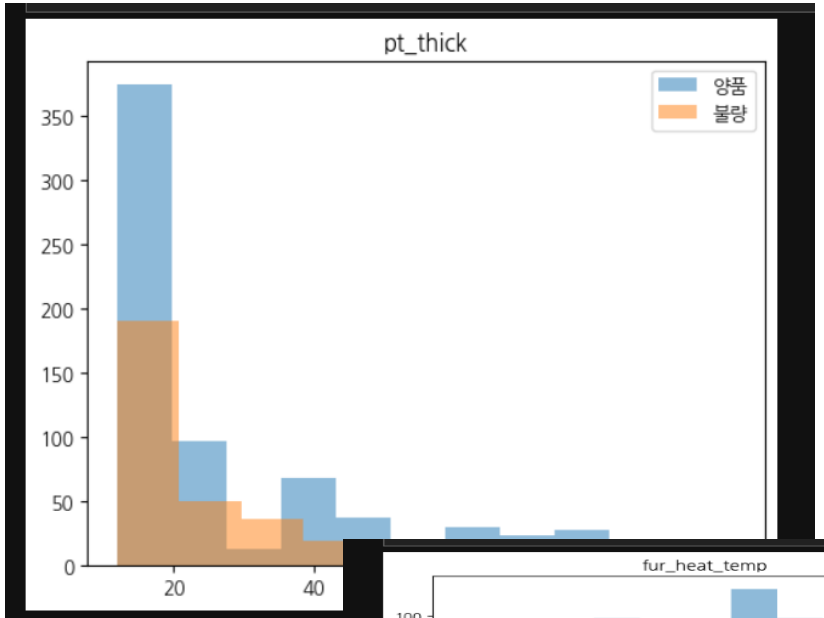


목표 변수인 scale 분포 현황 확인

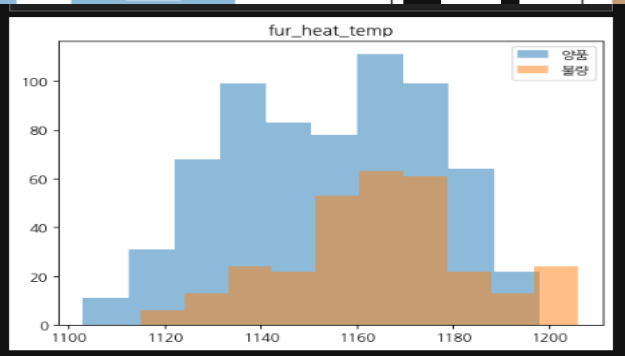


Project

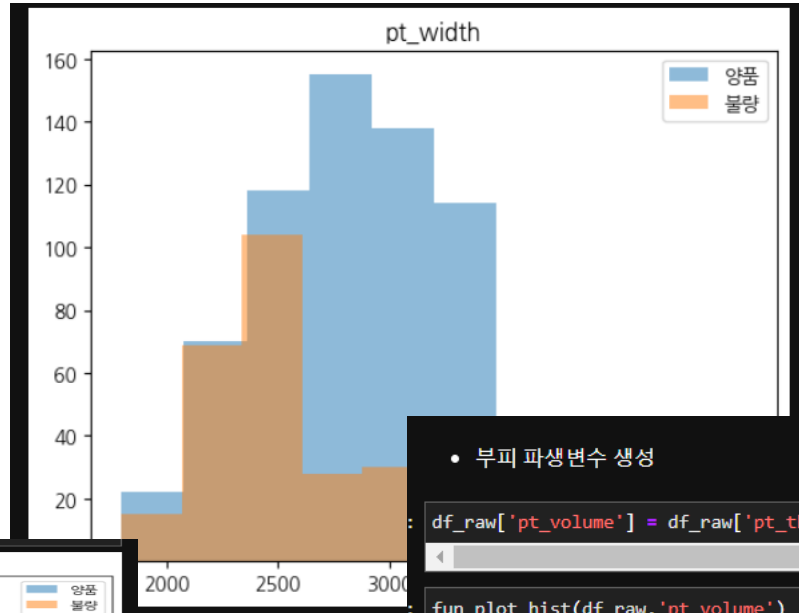
EDA (목표변수와 설명변수들의 분포 확인)



- 후판 두께가 10에서 20사이일때
- 이를 통해 대부분의 후판들은 얇



- 가열로의 온도가 1200도가 올라간다면 오로지 불량품만 있음을 확인 / 1200도의 온도로 올라가지 않도록 노력하는게 좋을 것 같음
- 1100도 부터 1115도까지는 양품만 나옴을 확인
- 상대적으로 적은 온도일 수록 불량품의 비율이 현저히 줄어듦 / 1150도 부터 불량률 급격히 증가함 / 고온으로 올릴시에는 물성이 변형되거나 소성이 일어나 제품의 강도가 감소 즉 불량품 생김
- 공정의 경우 일반적으로 온도와 생산속도는 trade off 관계를 가짐 / 온도와 생산 속도와의 상관관계를 확인할 필요가 있음

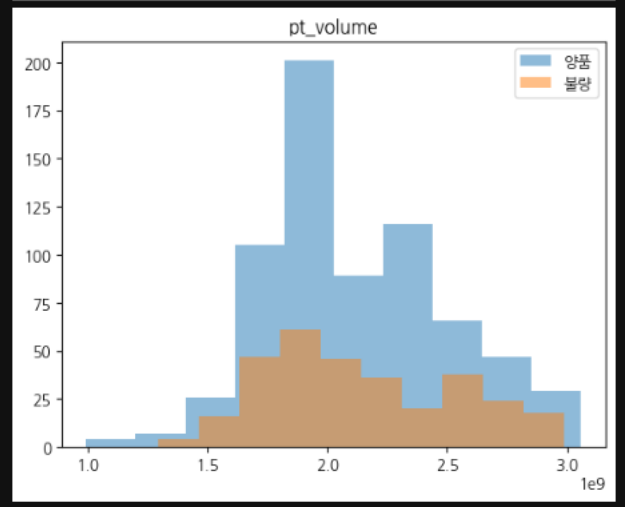


지시폭이 너무 얇거나 너무 두까
정규분포를 띄고 있으며, 적당히

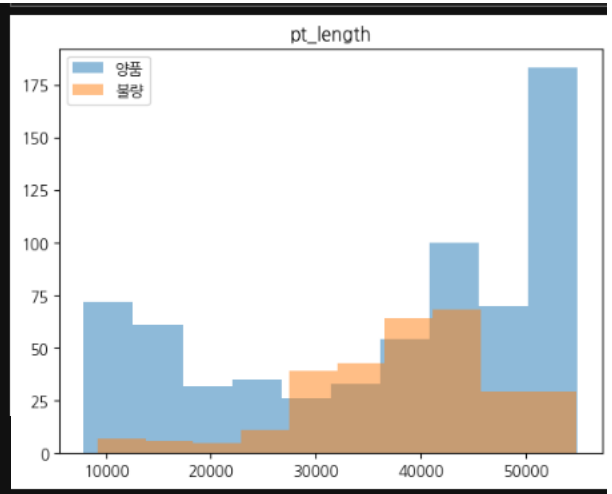
- 부피 파생변수 생성

```
df_raw['pt_volume'] = df_raw['pt_thick'] * df_raw['pt_w
```

```
fun_plot_hist(df_raw,'pt_volume')
```

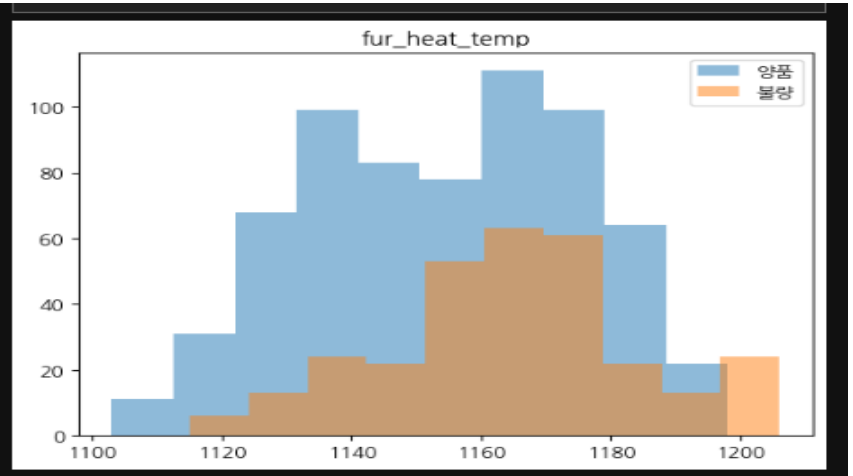


- 부피의 경우 중간의 부피를 가지고 있는 후판들이 제일 많이 생성되고 있으며 그만큼 양품의 비율도 높음을 확인하였다.



- 후판의 길이가 높을수록 양품의 비율이 높아짐을 확인하였다.
- 이를 토대로 부피의 파생변수를 만들어 scale과의 연관성을 확인해보고자 생각하였다.

EDA (목표변수와 설명변수들의 분포 확인)

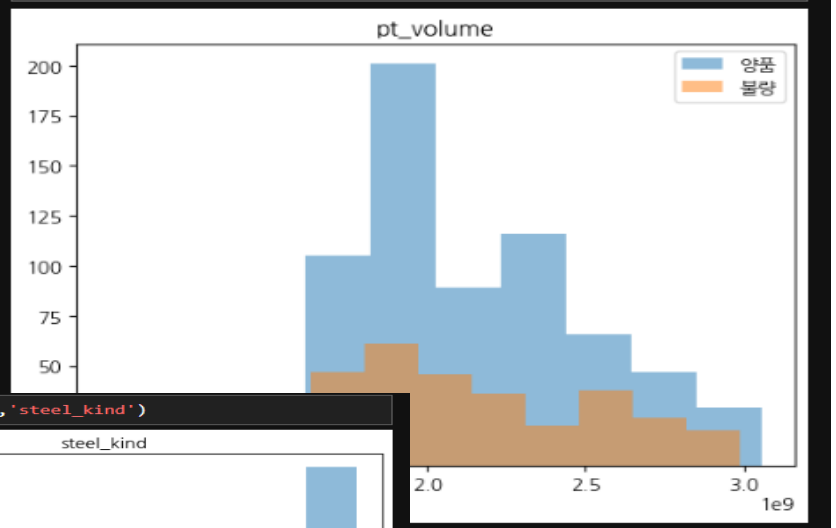


- 가열로의 온도가 1200도가 올라간다면 오로지 불량품만 있음을 확인 / 1200도의 온도로 올라가지 않도록 노력하는게 좋을 것 같음
- 1100도 부터 1115도까지는 양품만 나음을 확인
- 상대적으로 적은 온도일 수록 불량품의 비율이 줄어듦 / 1150도 부터 불량률 급격히 증가함 올릴시에는 물성이 변형되거나 소성이 일어나는 온도가 감소 즉 불량품 생김
- 공정의 경우 일반적으로 온도와 생산속도는 관계를 가짐 / 온도와 생산 속도와의 상관관계 요가 있음

- 부피 파생변수 생성

```
df_raw['pt_volume'] = df_raw['pt_thick'] * df_raw['pt_w
```

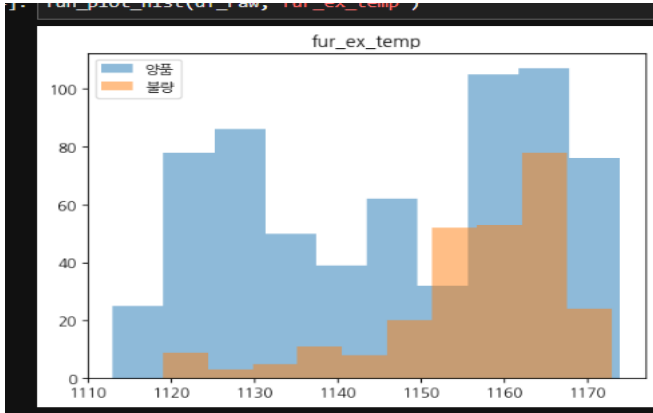
```
fun_plot_hist(df_raw,'pt_volume')
```



가지고 있는 후판들이 제일 큼 양품의 비율도 높음을 확

```
fun_plot_hist(df_raw,'steel_kind')
```

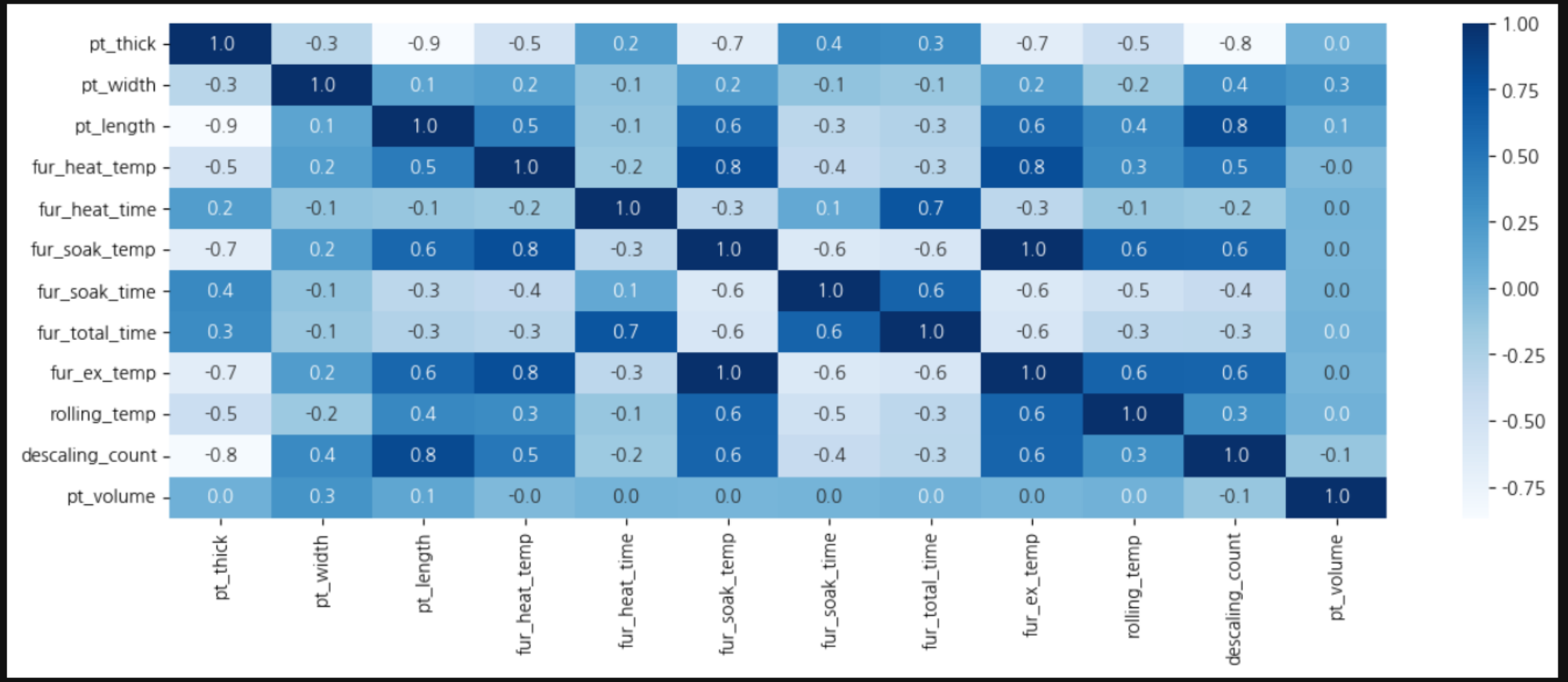
- T의 경우 가볍고 강한 소재인 대신 매우 비싸다. 그렇기 때문에 항공 우주, 의료 장비 등에 사용이 되고, 가격이 비싸 수요가 적은 것을 볼 수있다.
- 탄소의 경우 저렴하고 우수한 기계적 특성을 갖추어 있고 강도와 인성이 뛰어나기 때문에 건축물, 자동차, 기계류 등 다양한 제품에 사용된다. 그에따라서 수요도 많고 많이 생산하다보니 양품률과 불량률이 둘 다 높은 상황이다.



- 가열로 추출 온도가 올라갈수록 불량률 또한 상승함을 알 수 있다.
- 1140까지는 열을 최대한 유지하는 것이 불량률을 낮추는 데에 도움이 될것이다.

EDA (상관관계 확인)

112]: <Axes: >



- 플레이트의 특징들에 강한 음의 상관관계가 있음을 확인함.
- 온도는 온도끼리 양의 상관관계를 가짐을 확인함.
- 파생변수로 만들었던 부피의 경우 생각보다 다른 변수들과 상관관계를 띄고 있지 않음을 확인함.

EDA (영향을 주지 않을 것같은 인자 제거)

```
df_raw.drop(["descaling_count","plate_no","spec_country","fur_no","fur_input_row",'spec_long'],axis=1,inplace=True)
df_raw.head()
```

	scale	steel_kind	pt_thick	pt_width	pt_length	hsb	fur_heat_temp	fur_heat_time	fur_soak_temp	fur_soak_time	fur_total_time	fur_ex_temp	rolling_method	rolling_temp	work_group	pt_
3	양품	T	32	3700	15100	적용	1144	116	1133	59	259	1133	TMCP(온도제어)	934	1조	178
2	양품	T	32	3700	15100	적용	1144	122	1135	53	238	1135	TMCP(온도제어)	937	1조	178
5	양품	T	33	3600	19200	적용	1129	116	1121	55	258	1121	TMCP(온도제어)	889	1조	228
1	양품	T	33	3600	19200	적용	1152	125	1127	68	266	1127	TMCP(온도제어)	885	1조	228
0	양품	T	38	3100	13300	적용	1140	134	1128	48	246	1128	TMCP(온도제어)	873	1조	156

1. descaling_count와 scale의 분포를 확인해 보니 연관이 없다고 판단
2. plate_number 또한 연관이 없다고 판단
3. spec_country의 경우 나라에 비해 원하는 후판의 규격이 다를것이다. 그렇기 때문에 나라간의 불량률 차이가 생겨난다고 생각한다. 일본이 까다로운 규격을 사용한다고 불량률이 많지만 이는 우리 공정의 문제점을 알아 봐야 되는것이기 때문에 제거하고 고려한다.
4. fur_no의 1,2,3호 모두 불량률에 차이가 나지 않기 때문에 연관이 없다고 판단
5. fur_input_row 장입열은 불량률이 서로 비슷하므로 연관이 없다고 판단
6. spec_long 제품의 규격의 종류가 너무 많은 상황이므로 그 규격들을 전부 scale과의 관계를 볼 수 없다고 판단해 제거

로지스틱 회귀분석 모델

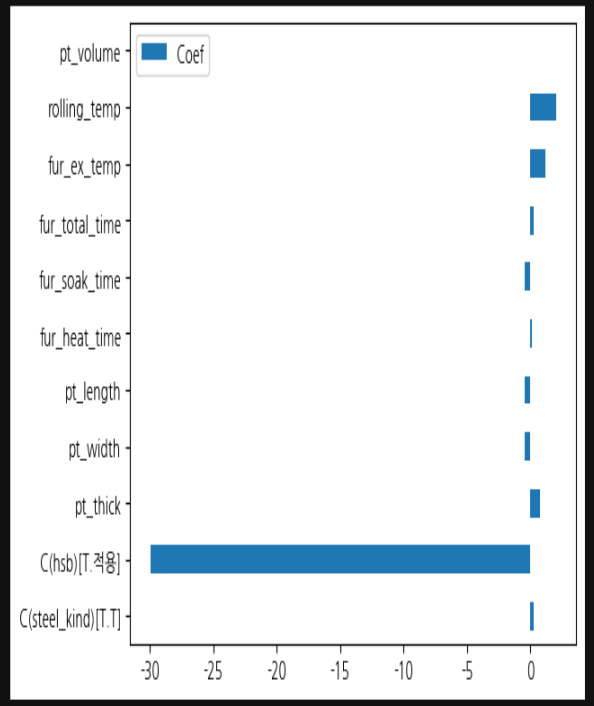
로지스틱 회귀모델 생성

```
135]: # 다중공전성 문제 때문에 상관계수가 높은 fur_heat_temp , fur_soek_temp 제외 후 로직모델 돌림

log_model = Logit.from_formula("scale ~ pt_thick + pt_width + pt_length  + \
                                fur_heat_time  + fur_soak_time + fur_total_time + \
                               fur_ex_temp + rolling_temp + pt_volume + C(steel_kind) + C(hsb)   ", df_train)

log_result = log_model.fit()

print(log_result.summary())
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.329164
         Iterations: 35
                           Logit Regression Results
==============================================================================
Dep. Variable:                  scale   No. Observations:                  695
Model:                          Logit   Df Residuals:                      683
Method:                           MLE   Df Model:                           11
Date:                Tue, 08 Aug 2023   Pseudo R-squ.:                  0.4772
Time:                        21:12:13   Log-Likelihood:                -228.77
converged:                      False   LL-Null:                       -437.61
Covariance Type:            nonrobust   LLR p-value:                1.073e-82
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept         27.9326   4.22e+04      0.001      0.999   -8.27e+04    8.28e+04
C(steel_kind)[T.T]   0.2574      1.049      0.245      0.806      -1.798       2.312
C(hsb)[T.적용]     -29.9469   4.22e+04     -0.001      0.999   -8.28e+04    8.27e+04
pt_thick          0.7397      0.683      1.082      0.279      -0.600       2.079
pt_width         -0.4609      0.262     -1.760      0.078      -0.974       0.052
pt_length        -0.4564      0.480     -0.950      0.342      -1.398       0.485
fur_heat_time     0.1435      0.253      0.568      0.570      -0.352       0.639
fur_soak_time    -0.4370      0.258     -1.696      0.090      -0.942       0.068
fur_total_time    0.2767      0.303      0.913      0.361      -0.317       0.871
fur_ex_temp       1.2086      0.259      4.659      0.000       0.700       1.717
rolling_temp      2.0269      0.355      5.707      0.000       1.331       2.723
pt_volume        -0.0012      0.220     -0.006      0.996      -0.433       0.430
==============================================================================
```

- 기본설정의 로지스틱 모델의 설명력은 58.7%이다.
- 유의수준 0.05보다 낮은 변수는 fur_ex_temp, rolling_temp가 존재한다. 즉 유의한 변수이다.

37]: <Axes: >



- hsb, rolling_temp, steel_kind, pt_length 순으로 scale 불량에 영향을 많이 끼치는것을 확인함

- hsb 과정을 "진행하지 않았을 때" scale 불량일 가능성이 높다.
- 강종이 Carbon Steel 이 아닌 "Tool Steel일 때" scale 불량일 가능성이 높다.
- 압연온도가 "높을수록" scale 불량일 가능성이 높다.
- 가열로 균열대 온도가 "높을수록" scale 불량일 가능성이 높다.

여기서hsb란?

후판 제조 과정에서 열처리를 수행하는 장비 또는 공정을 의미 스케일은 강재 표면에 생성되는 산화 철로 이루어진 층을 의미 / 강재를 고온에서 가공하거나 열처리하는 과정에서 생성 / 이 스케일은 강재의 표면 품질을 저하시키고 미세 구조에 영향을 줄 수 있기 때문에 제거가 필요 이러한 스케일을 제거 해주는 공정이 hsb hsb 진행시 표면 품질 개선, 생산성 향상, 제품 품질 향상등의 장점이 생김 즉 hsb는 양품의 후판을 생성하는데 필수적인 공정임

Random Forest

기본 옵션으로 랜덤포레스트 모델 생성

```
]: rf_uncustomized = RandomForestClassifier()
rf_uncustomized.fit(df_train_x,df_train_y)

#train data 설명력 (결정계수 확인)
print("score on training set :{:.3f}".format(rf_uncustomized.score(df_train_x,df_train_y)))

#test data 설명력 (결정계수 확인)
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 :0.955
```

- test data의 설명력이 높지만 train data의 설명력이 1이므로 과적합인 상황

```
]: estimator=RandomForestClassifier()

param_grid={'n_estimators':range(10,101,10),'max_depth':range(2,21,2),'min_samples_leaf':range(1,21)}

grid_rf=GridSearchCV(estimator,param_grid,scoring='accuracy',n_jobs=-1)
grid_rf.fit(df_train_x,df_train_y)
print('best estimator model:\n{}'.format(grid_rf.best_estimator_))
print('\nbest parameter:\n{}'.format(grid_rf.best_params_))
print('\nbest score:\n{}'.format(grid_rf.best_score_.round(3)))
```

```
best estimator model:
RandomForestClassifier(max_depth=4, n_estimators=90)

best parameter:
{'max_depth': 4, 'min_samples_leaf': 1, 'n_estimators': 90}

best score:
0.955
```

- 최종 모델 선정

GridSearchCV의 결과 min_samples_leaf = 1이 나왔지만 과소적합의 가능성이 있다고 생각해 3 결정

```
54]: rf_final=RandomForestClassifier(min_samples_leaf=3,max_depth=4,n_estimators=90)
rf_final.fit(df_train_x,df_train_y)
y_pred=rf_final.predict(df_test_x)

print('Accuracy on training set:{:.3f}'.format(rf_final.score(df_train_x,df_train_y)))

print('Accuracy on test set:{:.3f}'.format(rf_final.score(df_test_x,df_test_y)))

print('Confusion matrix:\n{}'.format(confusion_matrix(df_test_y,y_pred)))

print(classification_report(df_test_y,y_pred,digits=3))
```

```
Accuracy on training set:0.955
Accuracy on test set:0.965
Confusion matrix:
[[275   1]
 [ 13 109]]
              precision    recall  f1-score   support

           0      0.955     0.996     0.975       276
           1      0.991     0.893     0.940       122

    accuracy                          0.965       398
   macro avg      0.973     0.945     0.957       398
weighted avg      0.966     0.965     0.964       398
```

GridSearchCV를 이용해 최적 파라미터를 찾아냄
결과값의 min_sample_leaf = 1 이 나왔지만 과소적합의 가능성이 존재할 수 있다고 생각해 이를 방지하고자 3을 결정

KNN

KNN 모델

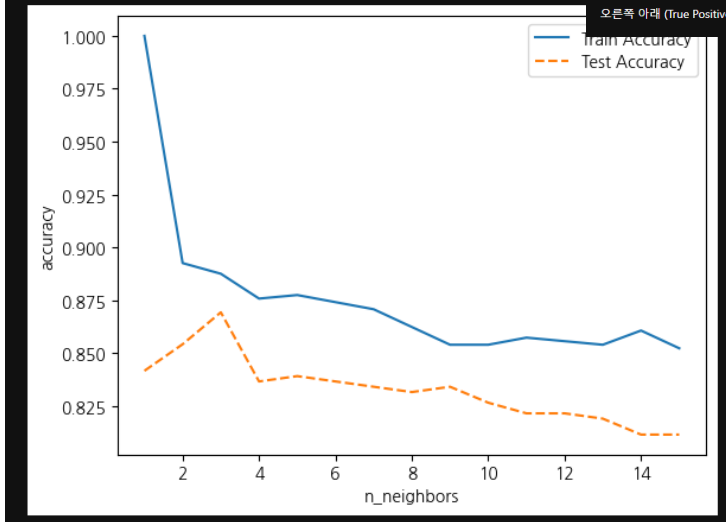
```
7]: # train 및 test 정확도 결과 저장용
train_accuracy = []; test_accuracy = []

para_n_neighbors = [i for i in range(1, 16)]

for v_n_neighbors in para_n_neighbors:
    knn = KNeighborsClassifier(n_neighbors=v_n_neighbors)
    knn.fit(df_train_x, df_train_y)
    train_accuracy.append(knn.score(df_train_x, df_train_y))
    test_accuracy.append(knn.score(df_test_x, df_test_y))

# 데이터 테이블로 저장
df_accuracy_neighbors = pd.DataFrame()
df_accuracy_neighbors["Neighbors"] = para_n_neighbors
df_accuracy_neighbors["TrainAccuracy"] = train_accuracy
df_accuracy_neighbors["TestAccuracy"] = test_accuracy
df_accuracy_neighbors.round(3)
```

7]:

	Neighbors	TrainAccuracy	TestAccuracy
0	1	1.000	0.842
1	2	0.893	0.854
2	3	0.888	0.869
3	4	0.876	0.837
4	5	0.878	0.839
5	6	0.874	0.837
6	7	0.871	0.834
7	8	0.862	0.832
8	9	0.854	0.834
9	10	0.854	0.827
10	11	0.857	0.822

```
]: # 정확도를 그래프로 표현
plt.plot(para_n_neighbors, train_accuracy, linestyle = "-", label
plt.plot(para_n_neighbors, test_accuracy, linestyle = "--", label
plt.ylabel("accuracy")
plt.xlabel("n_neighbors")
plt.legend()
```

```
]: <matplotlib.legend.Legend at 0x2057b8516d0>
```



- 이웃수가 증가할 수록 정확도는 낮아지는 방향으로 감
- 이웃의 3으로 설정하는 것이 train data의 정확도도 계속 떨어지지 않으며 test data의 정확도도 제일 높기 때문에 선택

최종 모델 선택

```
knn_final = KNeighborsClassifier(n_neighbors=4,weights="uniform",metric="manhattan")
knn_final.fit(df_train_x, df_train_y)

y_pred = knn_final.predict(df_test_x)

print("Accuracy on training set: {:.3f}".format(knn_final.score(df_train_x, df_train_y)))
print("Accuracy on test set: {:.3f}\n".format(knn_final.score(df_test_x, df_test_y)))
print("Confusion matrix: \n{}".format(confusion_matrix(df_test_y, y_pred)))
```

```
Accuracy on training set: 0.879
Accuracy on test set: 0.852

Confusion matrix:
[[267   9]
 [ 50  72]]
```

- train data에서 87.9%의 정확도를 가지고, test data에서는 85.2%의 정확도를 가짐. 정확도 차이가 크지 않아서 과적합(Overfitting) 문제가 크게 발생하지 않은 것으로 생각됨. 모델이 새로운 데이터에 대해서도 일정 수준의 일반화 능력을 가지고 있는 것으로 판단

왼쪽 위 (True Negative, TN): 267개의 샘플이 실제로 음성 클래스(0, 정상)이었고, 모델도 정확하게 음성으로 예측

오른쪽 위 (False Positive, FP): 9개의 샘플이 실제로 음성 클래스였지만 모델이 불량으로 잘못 예측

왼쪽 아래 (False Negative, FN): 50개의 샘플이 실제로 양성 클래스(1, 불량)였지만 모델이 정상으로 잘못 예측

오른쪽 아래 (True Positive, TP): 72개의 샘플이 실제로 양성 클래스였고, 모델도 정확하게 양성으로 예측

기본, 가중치, 거리 계산 방법을 변화해보며 최적의 결과값 도출

KNN

KNN 모델

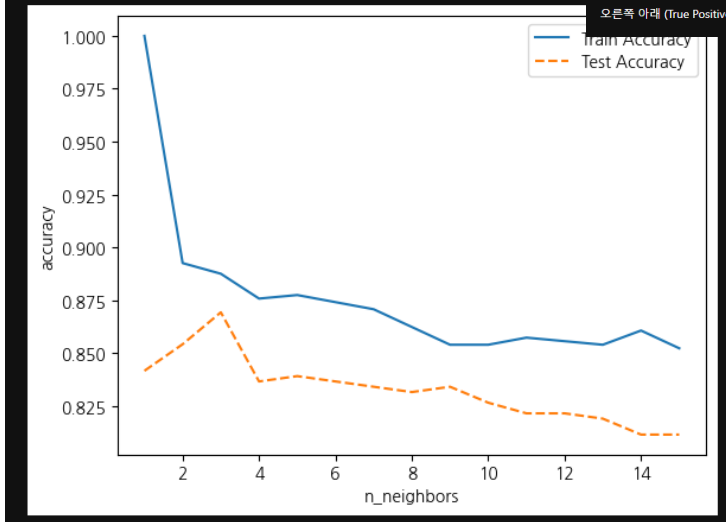
```
7]: # train 및 test 정확도 결과 저장용
train_accuracy = []; test_accuracy = []

para_n_neighbors = [i for i in range(1, 16)]

for v_n_neighbors in para_n_neighbors:
    knn = KNeighborsClassifier(n_neighbors=v_n_neighbors)
    knn.fit(df_train_x, df_train_y)
    train_accuracy.append(knn.score(df_train_x, df_train_y))
    test_accuracy.append(knn.score(df_test_x, df_test_y))

# 데이터 테이블로 저장
df_accuracy_neighbors = pd.DataFrame()
df_accuracy_neighbors["Neighbors"] = para_n_neighbors
df_accuracy_neighbors["TrainAccuracy"] = train_accuracy
df_accuracy_neighbors["TestAccuracy"] = test_accuracy
df_accuracy_neighbors.round(3)
```

7]:

	Neighbors	TrainAccuracy	TestAccuracy
0	1	1.000	0.842
1	2	0.893	0.854
2	3	0.888	0.869
3	4	0.876	0.837
4	5	0.878	0.839
5	6	0.874	0.837
6	7	0.871	0.834
7	8	0.862	0.832
8	9	0.854	0.834
9	10	0.854	0.827
10	11	0.857	0.822

```
]: # 정확도를 그래프로 표현
plt.plot(para_n_neighbors, train_accuracy, linestyle = "-", label
plt.plot(para_n_neighbors, test_accuracy, linestyle = "--", label
plt.ylabel("accuracy")
plt.xlabel("n_neighbors")
plt.legend()
```

```
]: <matplotlib.legend.Legend at 0x2057b8516d0>
```



- 이웃수가 증가할 수록 정확도는 낮아지는 방향으로 감
- 이웃의 3으로 설정하는 것이 train data의 정확도도 계속 떨어지지 않으며 test data의 정확도도 제일 높기 때문에 선택

최종 모델 선택

```
knn_final = KNeighborsClassifier(n_neighbors=4,weights="uniform",metric="manhattan")
knn_final.fit(df_train_x, df_train_y)

y_pred = knn_final.predict(df_test_x)

print("Accuracy on training set: {:.3f}".format(knn_final.score(df_train_x, df_train_y)))
print("Accuracy on test set: {:.3f}\n".format(knn_final.score(df_test_x, df_test_y)))
print("Confusion matrix: \n{}".format(confusion_matrix(df_test_y, y_pred)))
```

```
Accuracy on training set: 0.879
Accuracy on test set: 0.852

Confusion matrix:
[[267   9]
 [ 50  72]]
```

- train data에서 87.9%의 정확도를 가지고, test data에서는 85.2%의 정확도를 가짐. 정확도 차이가 크지 않아서 과적합(Overfitting) 문제가 크게 발생하지 않은 것으로 생각됨. 모델이 새로운 데이터에 대해서도 일정 수준의 일반화 능력을 가지고 있는 것으로 판단

왼쪽 위 (True Negative, TN): 267개의 샘플이 실제로 음성 클래스(0, 정상)이었고, 모델도 정확하게 음성으로 예측

오른쪽 위 (False Positive, FP): 9개의 샘플이 실제로 음성 클래스였지만 모델이 불량으로 잘못 예측

왼쪽 아래 (False Negative, FN): 50개의 샘플이 실제로 양성 클래스(1, 불량)였지만 모델이 정상으로 잘못 예측

오른쪽 아래 (True Positive, TP): 72개의 샘플이 실제로 양성 클래스였고, 모델도 정확하게 양성으로 예측

기본, 가중치, 거리 계산 방법을 변화해보며 최적의 결과값 도출

결론 및 대안 도출

Introduce project

1. 탄소는 저렴하고 우수한 특성을 가져 강도와 인성이 뛰어나 많이 사용되고 있을 것임. -> steel_kind의 그래프를 보며 확인
2. HSB는 불량률을 낮추기 위해 필수적으로 해야 하는 공정임. ->HSB의 그래프와 다양한 모델링의 중요 변수 결과에서 HSB는 반드시 들어감
3. 공정 온도가 높을수록 물질의 물성이 변형되거나 소성이 일어나 제품의 강도가 감소해 불량률이 높아질것임. -> 온도 관련 변수들의 그래프를 보면 높은 온도에서 불량률이 높음을 확인함
4. 공정의 경우 일반적으로 온도와 생산속도는 trade off관계를 가질것임. -> 정확한 결론을 도출해내지 못했음
5. 길이, 폭, 높이가 있으니 부피 변수를 만든다면 scale과의 연관성이 있을것임. -> 부피 변수는 scale과의 연관성이 거의 없음을 확인

결론 및 대안 도출

1.정확도 비교
Random Forest가 96.5%로 다른 모델에 비해 훨씬 정확도가 높다.
KNN과 로지스틱은 서로 정확도가 85%와 87%로 비슷하다.

2. pecision, recall 비교
Logistic Regression: 정밀도 ≈ 0.822 재현율 ≈ 0.706
로지스틱 회귀 모델의 경우 정밀도는 약 0.822로, 모델이 불량으로 예측한 것 중에서 약 82.2%가 실제로 불량

재현율은 약 0.706으로, 실제 불량 중에서 약 70.6%를 모델이 정확하게 불량으로 예측

Random Forest: 정밀도 ≈ 0.991 재현율 ≈ 0.893
랜덤 포레스트 모델의 경우 정밀도는 약 0.991로, 모델이 불량으로 예측한 것 중에서 약 99.1%가 실제로 불량
재현율은 약 0.893으로, 실제 불량 중에서 약 89.3%를 모델이 정확하게 불량으로 예측

k-Nearest Neighbors: 정밀도 ≈ 0.889 재현율 ≈ 0.590
KNN 모델의 경우 정밀도는 약 0.889로, 모델이 불량으로 예측한 것 중에서 약 88.9%가 실제로 불량

재현율은 약 0.590으로, 실제 불량 중에서 약 59.0%를 모델이 정확하게 불량으로 예측

Introduce project

3.모델들이 선정한 중요한 변수
Logistic Regression: hsb, rolling_temp, steel_kind, pt_length 순으로 scale 불량에 영향을 많이 끼침

Random Forest: rolling_temp, pt_width, fur_soak_temp ,hsb 순으로 scale 불량에 영향을 더 많이 끼치는것을 확인함

공통적인 중요한 변수로는 rolling_temp, hsb가 존재한다.

최종적으로 rolling_temp, pt_width, fur_soak_temp ,hsb, steel_kind, pt_length 가 scale 불량에 큰 영향을 끼치는 인자로 선정됨

- 따라서 공정 온도는 너무 높지 않도록 (1110도 이상) 잘 컨트롤 하는 방법을 찾아내면 불량률을 줄일 수 있다고 생각한다.
- 또한 hsb 공정을 적용해야 scale의 불량을 줄이는데 매우 큰 도움을 줄것이다. Hsb를 수행하는 공정 장비나 라인을 하나 더 만들어서 공정을 진행한다면 생산 시간 단축, 불량률 감소와 같은 긍정적인 영향을 가져다 줄 것이다.

Introduce project

배운점

- 후판 공정 데이터를 전처리하고 모델링하며 얻은 경험은 저의 진로인 반도체 제조업 분야에도 잘 활용될 수 있다고 생각한다.
- 반도체 제조업에 한정되지 않고 다양한 제조 분야에서도 활용될 것이라고 생각한다.
- 다양한 제조 분야에서 공정의 유사성을 파악하고 데이터 분석 및 모델링 기술을 적용하여 생산 프로세스를 최적화하고 효율성을 높일 수 있을 역량을 키웠다.
- 직접 포스코의 데이터를 이용해 데이터 분석을 할 수 있는 경험은 어딜 가도 하지 못할 귀한 경험이라고 생각한다.
- 이론적으로 배웠을 때는 직관적으로 다가오지 않았지만 실습과 종합실습을 겪으며 내 지식이 되어가고 있다.
- 추후 진행할 프로젝트를 통해 완벽하게 이해할 수 있을 것 이라고 기대해본다.

End of Document