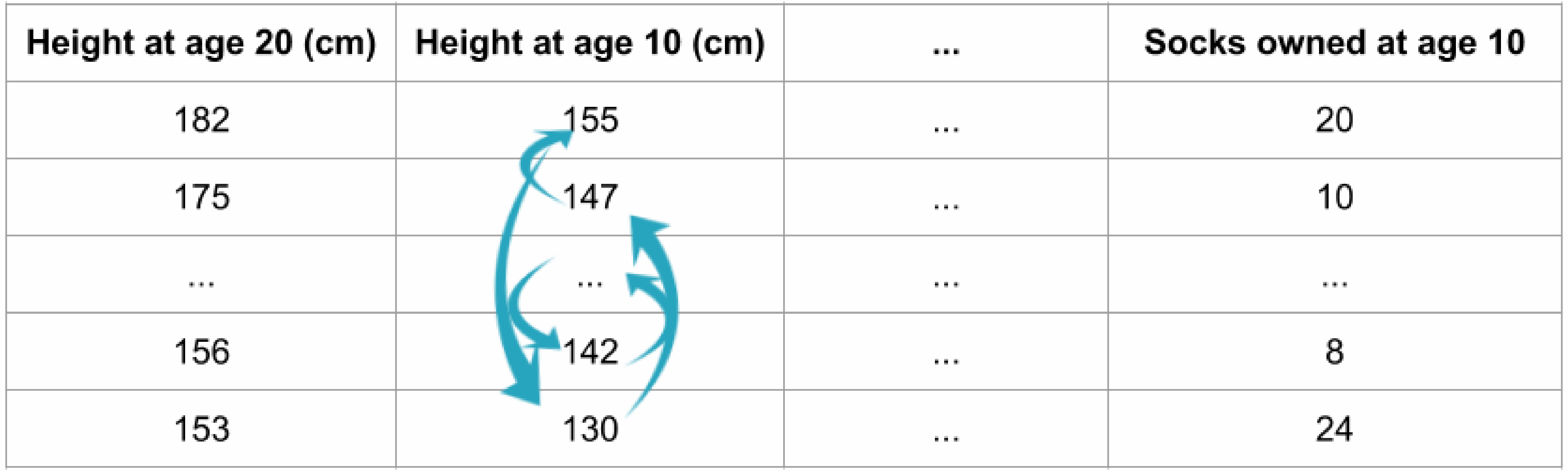
Permutation Importance

Permutation Importance

모델 예측에 가장 큰 영향을 미치는 Feature 를 파악하는 방법

특정 feature 값을 **무작위로 섞어서** 모델의 성능이 얼마나 떨어지는지를 보고, 그 **feature가 얼마나 중요한지를 측정**함



Height at age 20 (cm)	Height at age 10 (cm)	...	Socks owned at age 10
182	155	...	20
175	147	...	10
...	...	...	...
156	142	...	8
153	130	...	24

Permutation Importance

1. 모델을 학습시킨 후, **validation/test set에 대해 정확도를 측정**

2. 한 feature의 값을 **무작위로 섞음**
→ feature와 target 간의 관계를 끊음

3. 다시 예측을 해보고 **모델 성능이 얼마나 나빠졌는지를 측정**

4. 성능이 많이 떨어지면 → **중요한 feature**
성능변화가 거의 없으면 → **덜 중요한 feature**

Permutation Importance

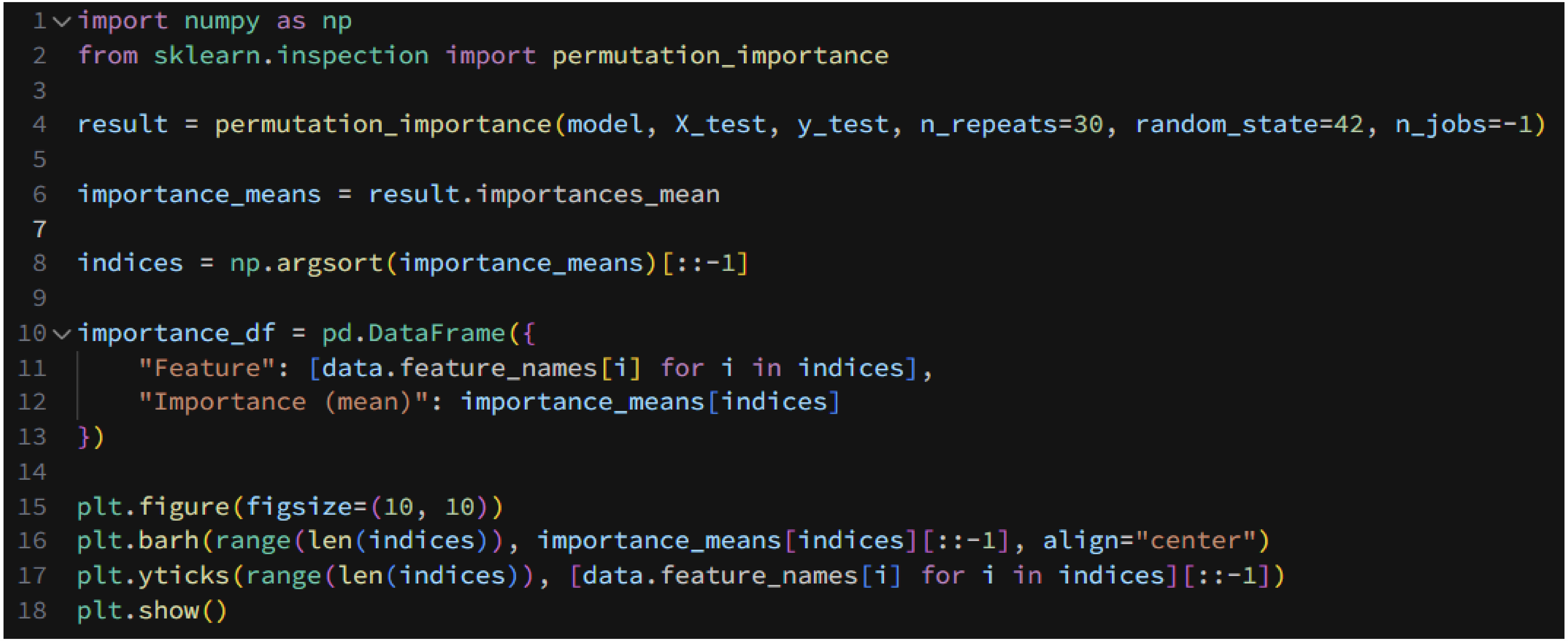
$$\text{Importance}(X_i) = \text{Score}_{\text{original}} - \text{Score}_{\text{shuffled on } X_i}$$

Ex) Score_o = 0.86

중요한 feature : Score_s = 0.12
덜 중요한 feature : Score_s = 0.82

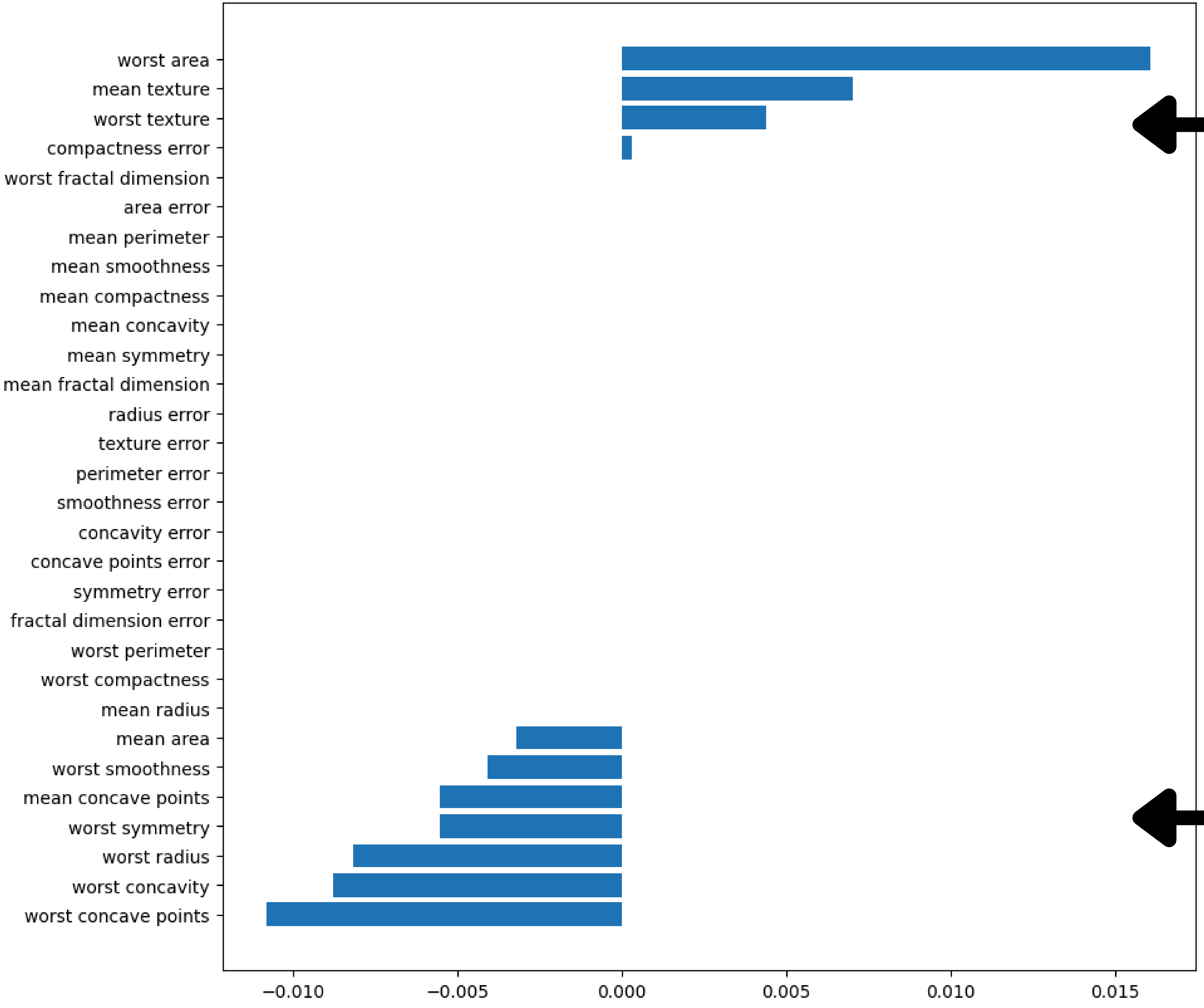
중요한 feature의 Importance : 0.74
덜 중요한 feature의 Importance : 0.04

Permutation Importance



```python
import numpy as np
from sklearn.inspection import permutation_importance

result = permutation_importance(model, X_test, y_test, n_repeats=30, random_state=42, n_jobs=-1)

importance_means = result.importances_mean

indices = np.argsort(importance_means)[::-1]

importance_df = pd.DataFrame({
    "Feature": [data.feature_names[i] for i in indices],
    "Importance (mean)": importance_means[indices]
})

plt.figure(figsize=(10, 10))
plt.barh(range(len(indices)), importance_means[indices][::-1], align="center")
plt.yticks(range(len(indices)), [data.feature_names[i] for i in indices][::-1])
plt.show()
```

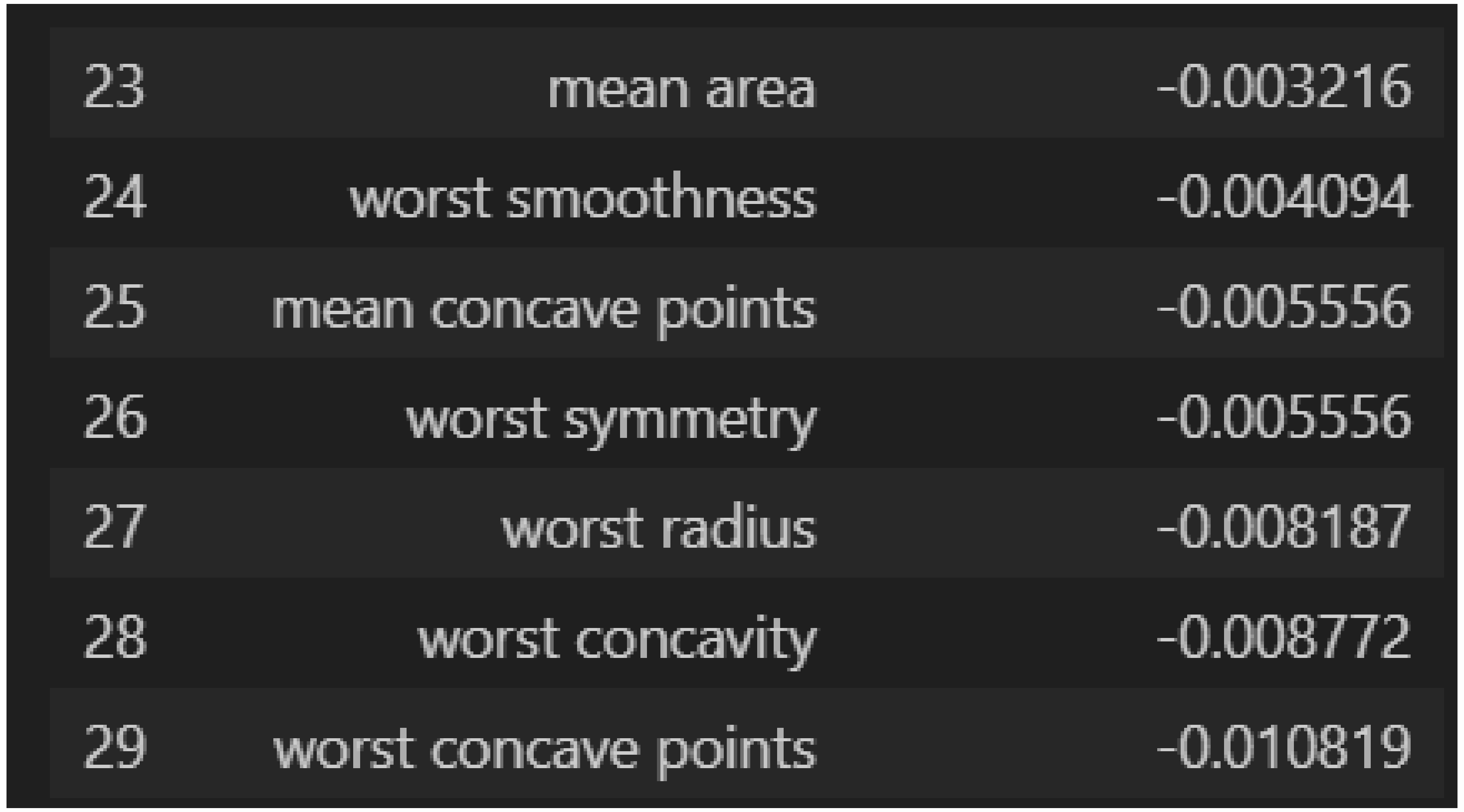
Permutation Importance



모델 성능에 영향을 많이 줌

모델 성능에 영향을 많이 안줌

Permutation Importance



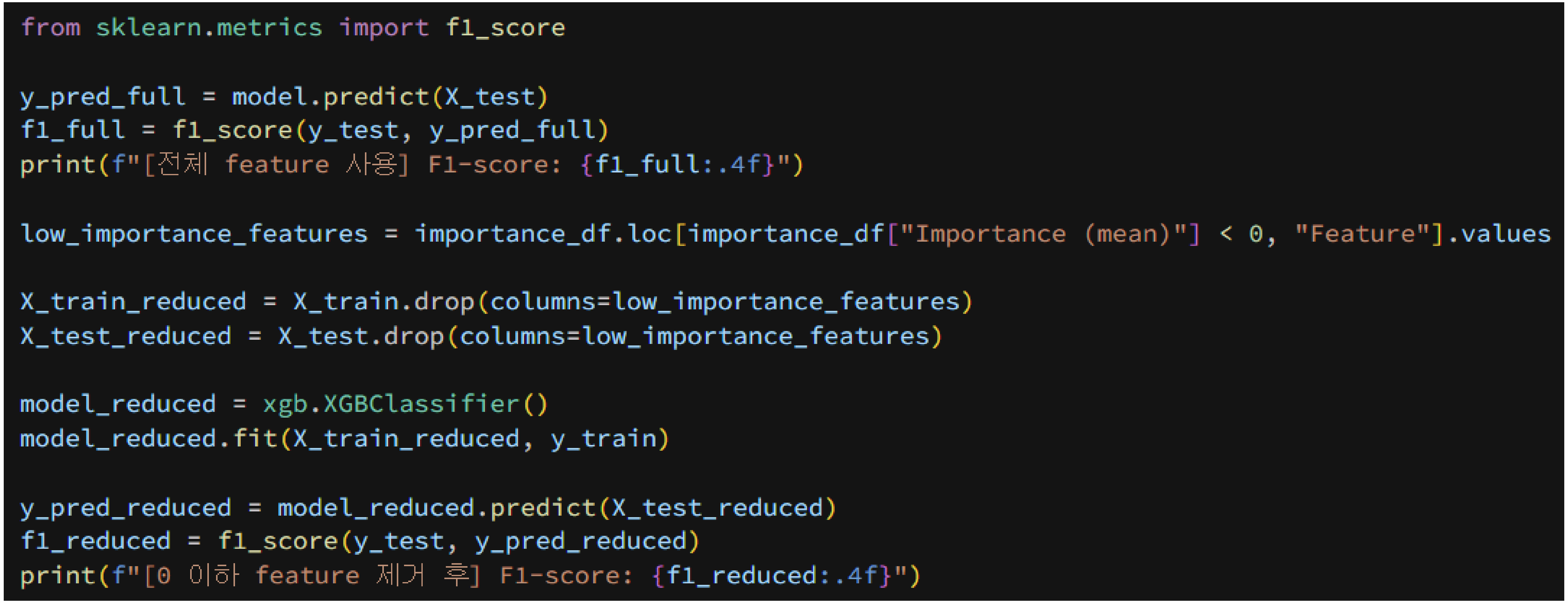
23	mean area	-0.003216
24	worst smoothness	-0.004094
25	mean concave points	-0.005556
26	worst symmetry	-0.005556
27	worst radius	-0.008187
28	worst concavity	-0.008772
29	worst concave points	-0.010819

Importance가 **음수**인 경우

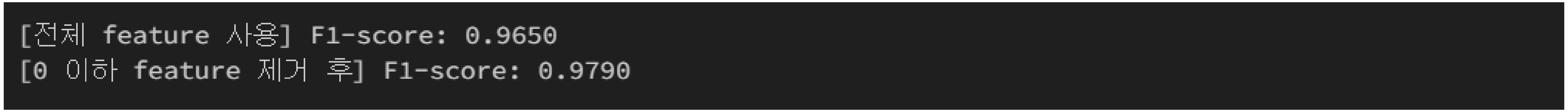
삭제 권장

→ **모델에 noise를 주는 feature**

Permutation Importance



```python
from sklearn.metrics import f1_score

y_pred_full = model.predict(X_test)
f1_full = f1_score(y_test, y_pred_full)
print(f"[전체 feature 사용] F1-score: {f1_full:.4f}")

low_importance_features = importance_df.loc[importance_df["Importance (mean)"] < 0, "Feature"].values

X_train_reduced = X_train.drop(columns=low_importance_features)
X_test_reduced = X_test.drop(columns=low_importance_features)

model_reduced = xgb.XGBClassifier()
model_reduced.fit(X_train_reduced, y_train)

y_pred_reduced = model_reduced.predict(X_test_reduced)
f1_reduced = f1_score(y_test, y_pred_reduced)
print(f"[0 이하 feature 제거 후] F1-score: {f1_reduced:.4f}")
```



```
[전체 feature 사용] F1-score: 0.9650
[0 이하 feature 제거 후] F1-score: 0.9790
```