Feature Selection

데이터 분석 스터디 2주차

Content

1. Correlation

2. Mutual Imformation

3. Shap value

4. Permutation Importance

Feature Selection

Feature Selection

데이터의 특성 중에서 **가장 중요하게 생각되는 변수를 선택**하는 것

Goal : 모델 훈련 과정에서 **가장 유용한 특성**을 선택하는 것

장점

- **차원의 저주**를 줄임
- 모델의 **성능을 높임**
- **과적합 방지**
- 데이터 분석에서 **중요하지 않은 특성 제거**

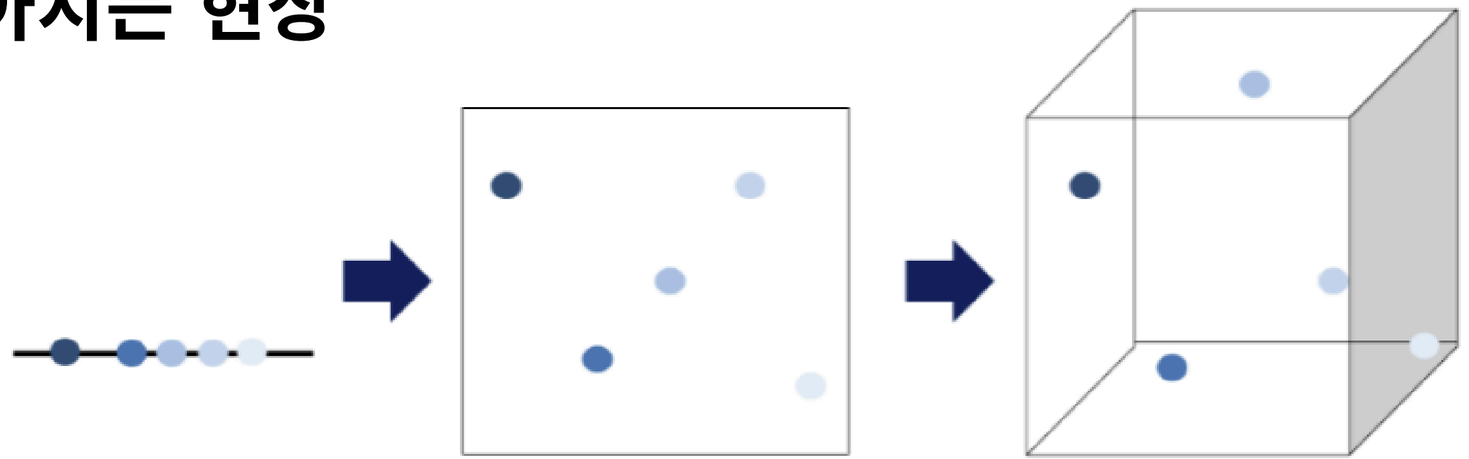
Feature Selection

차원의 저주

차원이 증가하면서 **문제공간이 지수적으로 커지는** 현상

- 용량이 커진다 → **필요한 메모리 큼**
- 데이터보다 변수 수가 많아짐 → **모델의 성능 저하**

차원이 커질 수록 공간이 많이 필요하면서 사용할 수 있는 **정보량은 상대적으로 작아지는 현상**

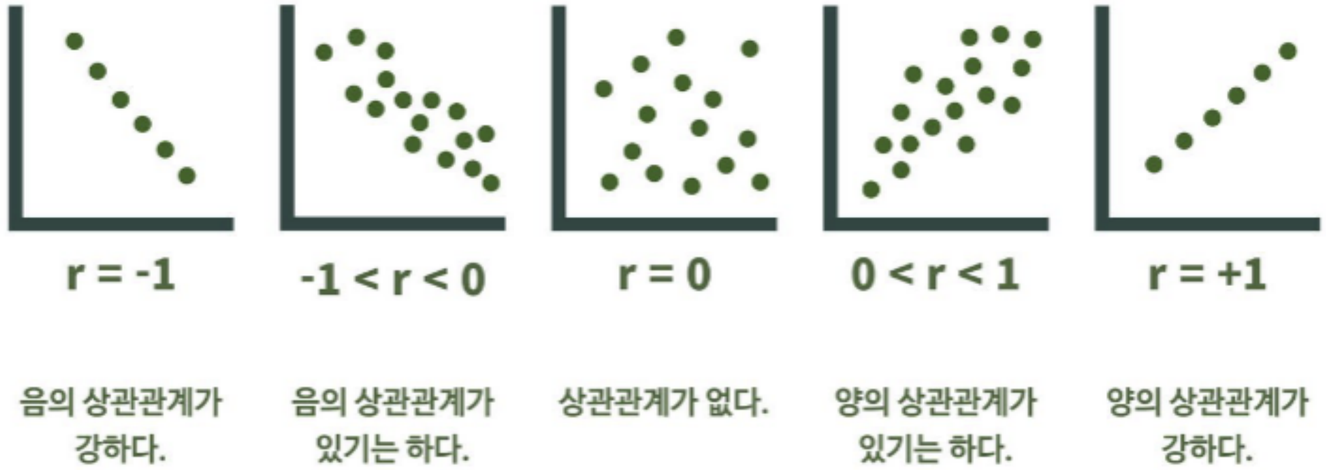


Correlation

상관계수(Correlation)

두 변수 사이에 관계를 나타내는 수치

- [-1, +1] 사이의 값을 가짐
- -1에 가까울수록 **음의 상관관계**
- +1에 가까울수록 **양의 상관관계**
- 0은 **관계가 없음**



Correlation

장점

- 간단하고 직관적인 방법
- 변수 간 관계 파악이 쉬움
- 계산이 단순

단점

- 비선형 관계 파악이 어려움
- 연관성을 볼 뿐, 인과관계를 설명하지 않음
- 결측치나 이상치에 민감하게 반응

Correlation

$$corr(A,B) = \frac{Cov(A,B)}{\sqrt{Var(A) \cdot Var(B)}}$$

$$Cov(A,B) = \frac{1}{n-1}\sum_{i=1}^{n}(A_i - \bar{A})(B_i - \bar{B})$$

$$Var(A) = \frac{1}{n-1}\sum_{i=1}^{n}(A_i - \bar{A})^2$$

표준화 ↓

$$corr(A,B) = \frac{Cov(A,B)}{\sqrt{Var(A)} \cdot \sqrt{Var(B)}} = \frac{\sum_{i=1}^{n}(A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum_{i=1}^{n}(A_i - \bar{A})^2} \cdot \sqrt{\sum_{i=1}^{n}(B_i - \bar{B})^2}}$$

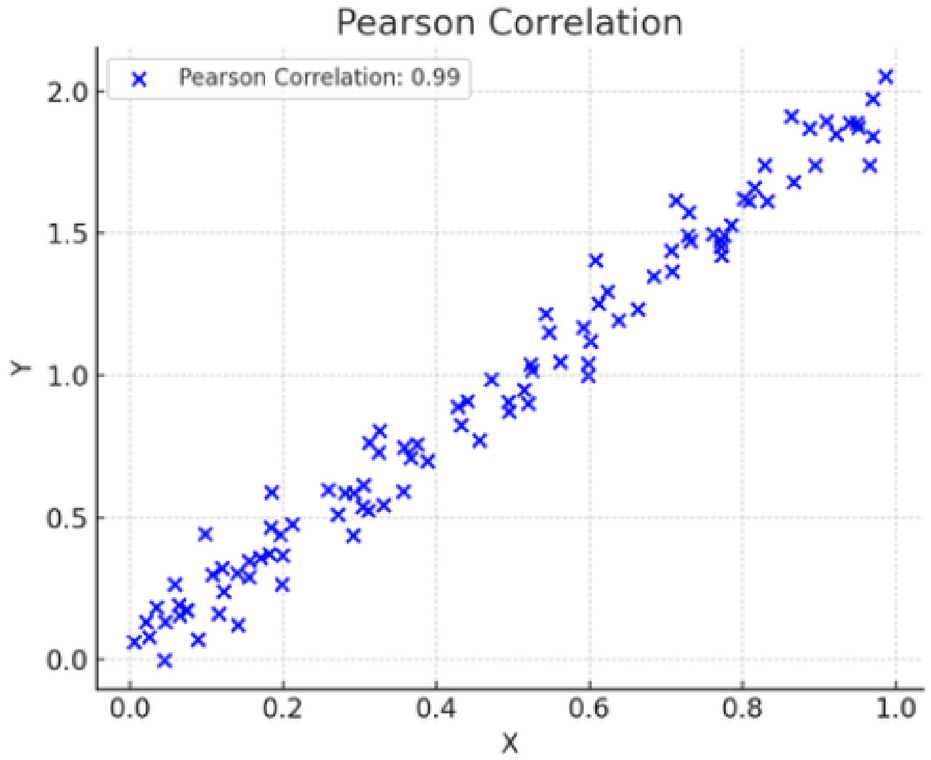
Correlation

이걸로 뭐하는거임?

차원의 저주를 해결 → **차원을 줄여야 함.**

비슷한 feature가 있으면 **두 feature 중 선택**

하나는 Drop



Correlation

x1	X2
3	6
4	9
8	21
2	6
7	8
2	3
4	5

$\overline{x_1} = 4.28$ $\overline{x_2} = 8.28$

≈ 4 ≈ 8

Correlation

x1	X2
3	6
4	9
8	21
2	6
7	8
2	3
4	5

$\overline{x_1} = 4.28 \approx 4$ $\overline{x_2} = 8.28 \approx 8$

X'1	X'2	X'1*x'2	∑
-1	-2	2	
0	1	0	
4	13	52	
-2	-2	4	68
3	0	0	
-2	-5	10	
0	-3	0	

Correlation

x1	X2
3	6
4	9
8	21
2	6
7	8
2	3
4	5

$\overline{x_1} = 4.28 \approx 4$ $\overline{x_2} = 8.28 \approx 8$

X'1	X'2	X'1*x'2	$\sum$
-1	-2	2	68
0	1	0	
4	13	52	
-2	-2	4	
3	0	0	
-2	-5	10	
0	-3	0	

A=(X'1)^2	B=(X'2)^2	$\sum A * \sum B$	$\sqrt{}$A*B
1	4	34*212= 7208	84.89
0	1		
16	169		
4	4		
9	0		
4	25		
0	9		

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} = \frac{68}{84.89} = 0.80$$

Bostion dataset (보스턴 주택 가격 데이터)

```
1  import statsmodels.api as sm
2
3  boston = sm.datasets.get_rdataset("Boston", "MASS").data
4  print(boston.head())
```

✓ 1.6s

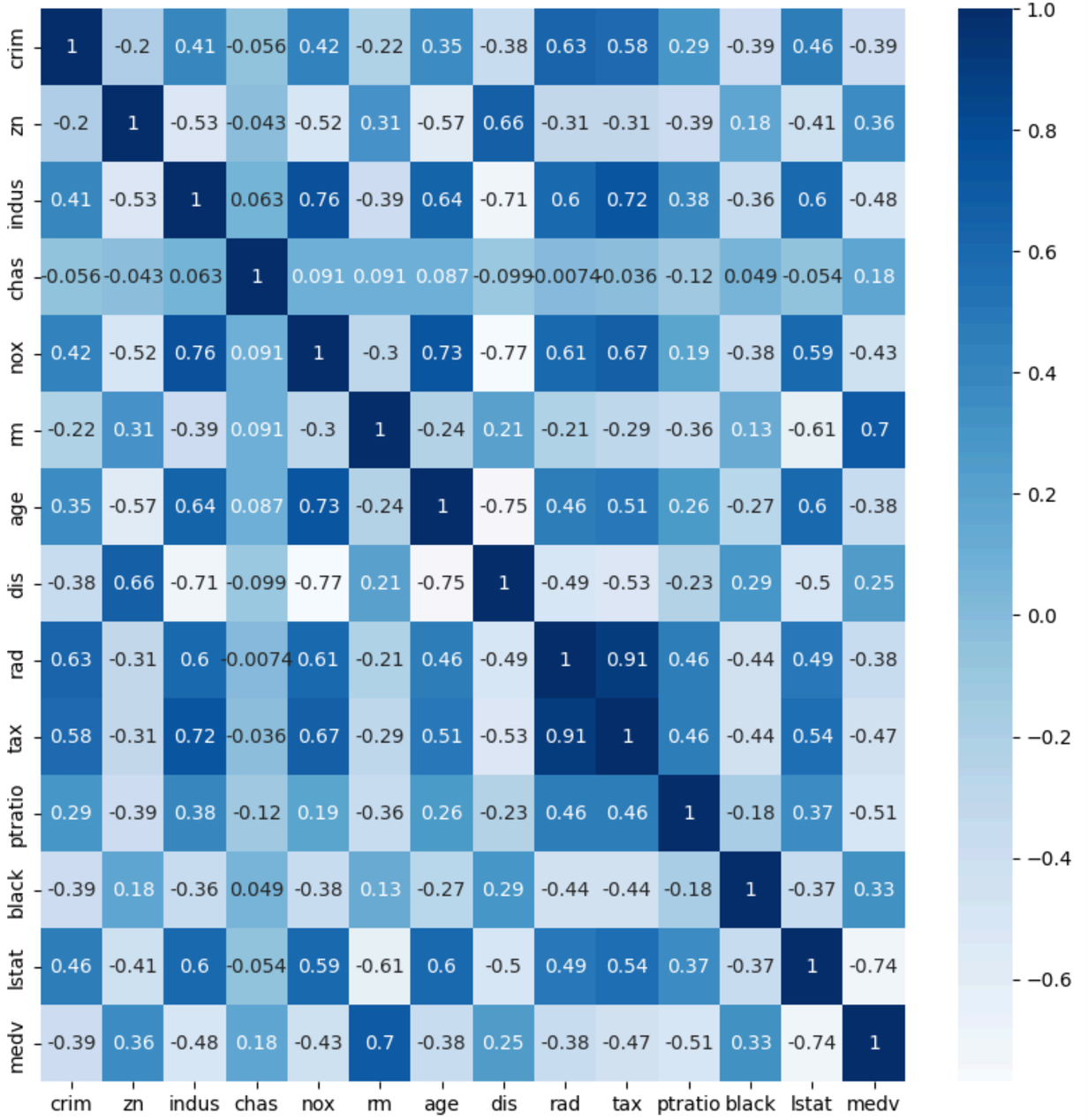
Bostion dataset (보스턴 주택 가격 데이터)

[01]	CRIM	자치시(town) 별 1인당 범죄율
[02]	ZN	25,000 평방피트를 초과하는 거주지역의 비율
[03]	INDUS	비소매상업지역이 점유하고 있는 토지의 비율
[04]	CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
[05]	NOX	10ppm 당 농축 일산화질소
[06]	RM	주택 1가구당 평균 방의 개수
[07]	AGE	1940년 이전에 건축된 소유주택의 비율
[08]	DIS	5개의 보스턴 직업센터까지의 접근성 지수
[09]	RAD	방사형 도로까지의 접근성 지수
[10]	TAX	10,000 달러 당 재산세율
[11]	PTRATIO	자치시(town)별 학생/교사 비율
[12]	B	1000(Bk-0.63)^2, 여기서 Bk는 자치시별 흑인의 비율을 말함.
[13]	LSTAT	모집단의 하위계층의 비율(%)
[14]	MEDV	본인 소유의 주택가격(중앙값) (단위: $1,000)

Bostion dataset (보스턴 주택 가격 데이터)

```
import matplotlib.pyplot as plt
import seaborn as sns

cor = boston.corr()

plt.figure(figsize=(10, 10))
sns.heatmap(cor, annot = True, cmap=plt.cm.Blues)
plt.show()
```

Correlation



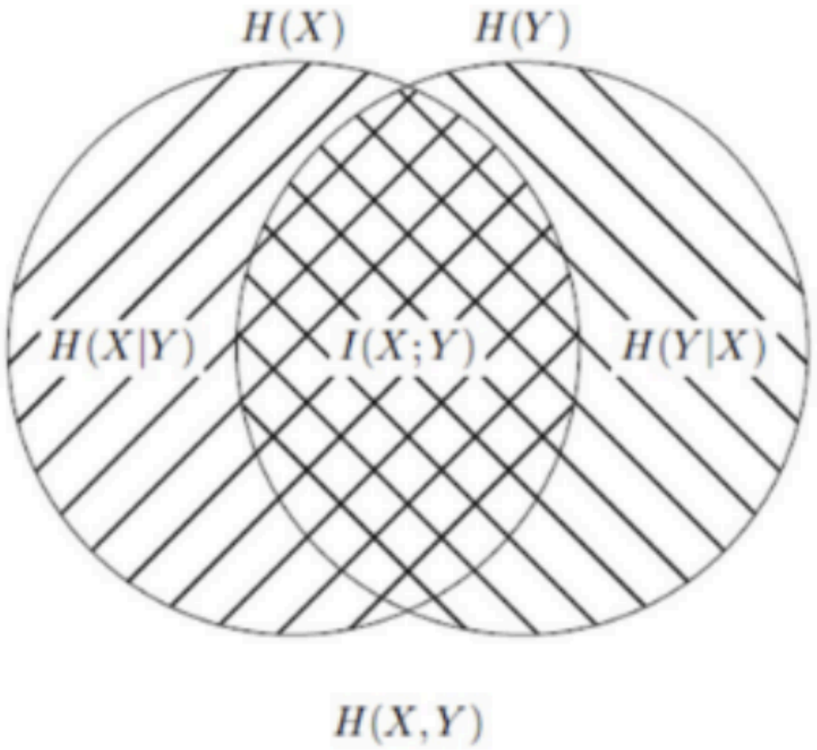
Mutual Imformation

상호정보량

두 확률 변수 간 의존성을 나타내는 척도

- **두 변수 간 의존성이 클 때** 해당 변수를 선택

$$I(X;Y) = H(X) + H(Y) - H(X, Y)$$
$$= H(X) - H(X|Y)$$
$$= H(Y) - H(Y|X)$$



Mutual Imformation

Entropy(엔트로피)
한 변수의 불확실성
(= 놀라움의 정도)

$$H(X) = -\sum_{x} P(x) \log P(x)$$

Joint Entropy
두 변수를 동시에 아는데
필요한 정보량

$$H(X,Y) = -\sum_{x,y} P(x,y) \log P(x,y)$$

Conditional Entropy
Y를 알고 있을 때 X의
남은 불확실성

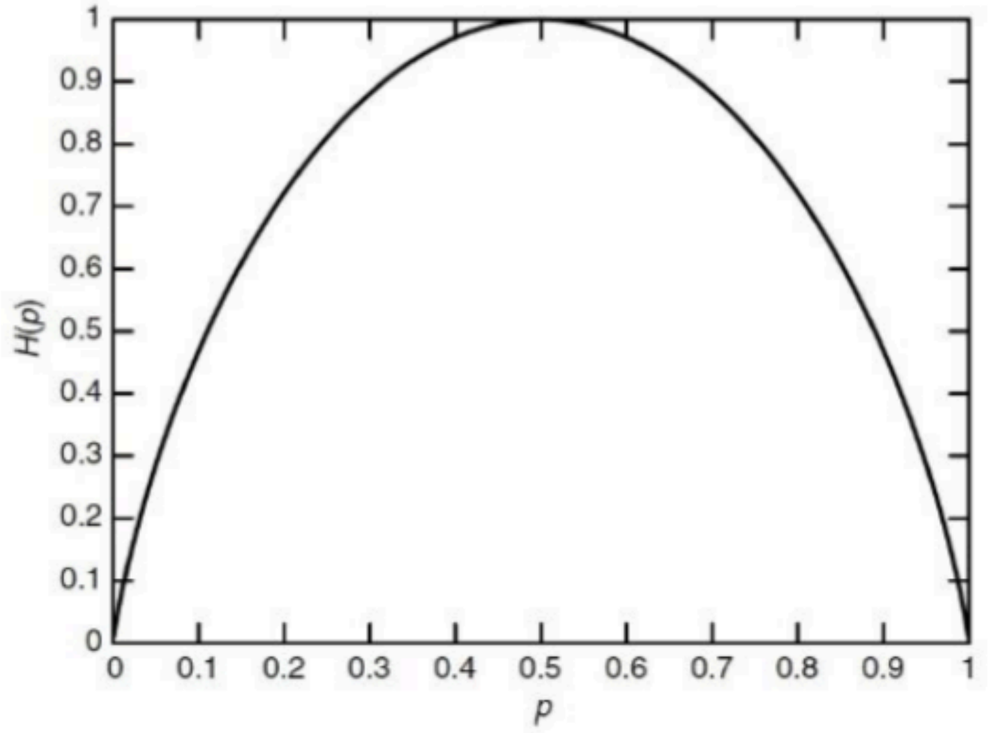
$$H(X|Y) = -\sum_{x,y} P(x,y) \log P(x|y)$$

Mutual Imformation

Ex) 해가 동쪽에서 뜬다. → p(동쪽) = 1.0

해가 서쪽에서 뜬다. → p(서쪽) = 0.0

확실한 사건!!!!! = 정보량이 없음 = 불확실하지 않음



Mutual Imformation

상호정보량을 이용한 변수 선택 과정

1. **각 feature와 target 간의 상호정보량** 계산

2. 계산된 상호정보량을 **기준으로 정렬**

3. **임계값을 기준**으로 **상위 순위의 특성을** 선택

Mutual Imformation

장점

- 다른 feature selection보다 정확함
- 비선형적 관계도 분석 가능
- 희소한 데이터도 적용 가능

단점

- 계산 비용이 매우 큼

Mutual Imformation

월(Month)	결제 패턴(Pattern)
1	A
2	B
1	C
3	A
2	B
3	C

Mutual Imformation

	1	2	3	주변빈도
A	1	0	1	2
B	0	2	0	2
C	1	0	1	2
주변빈도	2	2	2	6

	1	2	3	
A	P(1,A)	P(2,A)	P(3,A)	P(A)
B	P(1,B)	P(2,B)	P(3,B)	P(B)
C	P(1,C)	P(2,C)	P(3,C)	P(C)
	P(1)	P(2)	P(3)	

	1	2	3	
A	1/6	0	1/6	1/3
B	0	1/3	0	1/3
C	1/6	0	1/6	1/3
	1/3	1/3	1/3	

Mutual Imformation

	1	2	3	
A	1/6	0	1/6	1/3
B	0	1/3	0	1/3
C	1/6	0	1/6	1/3
	1/3	1/3	1/3	

$$I(\text{Month, Pattern}) = P(1,A)\log_2\left(\frac{P(1,A)}{(P(1)*P(A))}\right) + P(1,C)\log_2\left(\frac{P(1,C)}{(P(1)*P(C))}\right) + P(2,B)\log_2\left(\frac{P(2,B)}{(P(2)*P(B))}\right)$$
$$+P(3,A)\log_2\left(\frac{P(3,A)}{(P(3)*P(A))}\right) + P(3,C)\log_2\left(\frac{P(3,C)}{(P(3)*P(C))}\right)$$

$$=0.097$$

Mutual Imformation

유방암 데이터

```
1  from sklearn.datasets import load_breast_cancer
2
3  data = load_breast_cancer()
4  X = data.data
5  y = data.target
```

0.4s

Mutual Imformation

유방암 데이터

```python
from sklearn.feature_selection import mutual_info_classif, SelectKBest

# k개의 feature 선택
k = 10

# 상호정보량 계산
mi = mutual_info_classif(X, y)

# k개의 feature 선택
selector = SelectKBest(mutual_info_classif, k=k)
X_new = selector.fit_transform(X, y)

# 상호정보량이 가장 높은 feature 10개 출력
topk_indices = mi.argsort()[::-1][:10]
topk_features = data.feature_names[topk_indices]
print("Top {} Features:".format(k))

for feature in topk_features:
    print("- {}".format(feature))
```

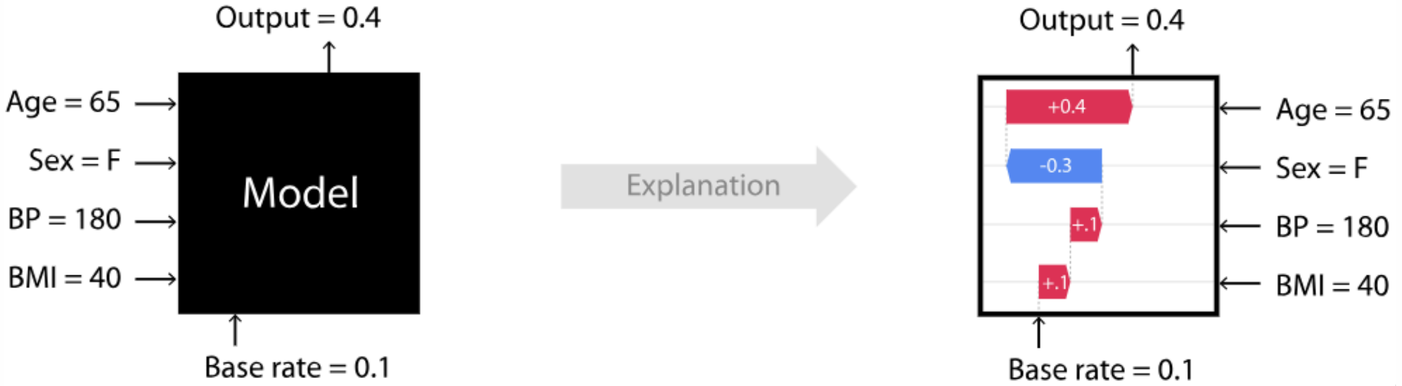
Shap value

SHAP(SHapley Additive exPlanations)

게임이론의 shapley value를 기반으로 한 모델 해석 기법

핵심 아이디어 : 모델 예측을 각 feature의 공정한 기여도로 분해하자!

SHAP



Shap value

Ex) 정훈, 영빈, 소연이 대회에서 500만원을 상금으로 받음.
→ 상금을 어떻게 배분할거냐? 누가 얼마씩 받아야 공평한가?

EDA → 데이터 전처리 → 모델링

순서	기여도 계산	정훈의 기여도
정훈 → 영빈 → 소연	정훈이의 EDA 덕분에 30점을 받음	**30**
영빈 → 정훈 → 소연	영빈이의 EDA는 40점을 받았고 정훈이의 데이터 전처리로 40점을 더 받음	**40**
영빈 → 소연 → 정훈	영빈이의 EDA와 소연이의 데이터 전처리로 90점을 받고 정훈이의 모델링으로 10점을 받음	**10**

Shap value

장점

- 정확한 기여도 계산
- 어느 모델이든 사용 가능
- 직관적으로 해석 가능하고 시각화 도구를 제공

단점

- 계산 비용이 매우 큼
- 딥러닝에서는 정확하지 않을 수 있음
- 대규모 데이터에서는 느림

Shap value

pip install shap xgboost

```
import shap
import xgboost as xgb
import pandas as pd
from sklearn.model_selection import train_test_split

data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Shap value

```
1  model = xgb.XGBClassifier()
2  model.fit(X_train, y_train)
```

✓ 0.0s

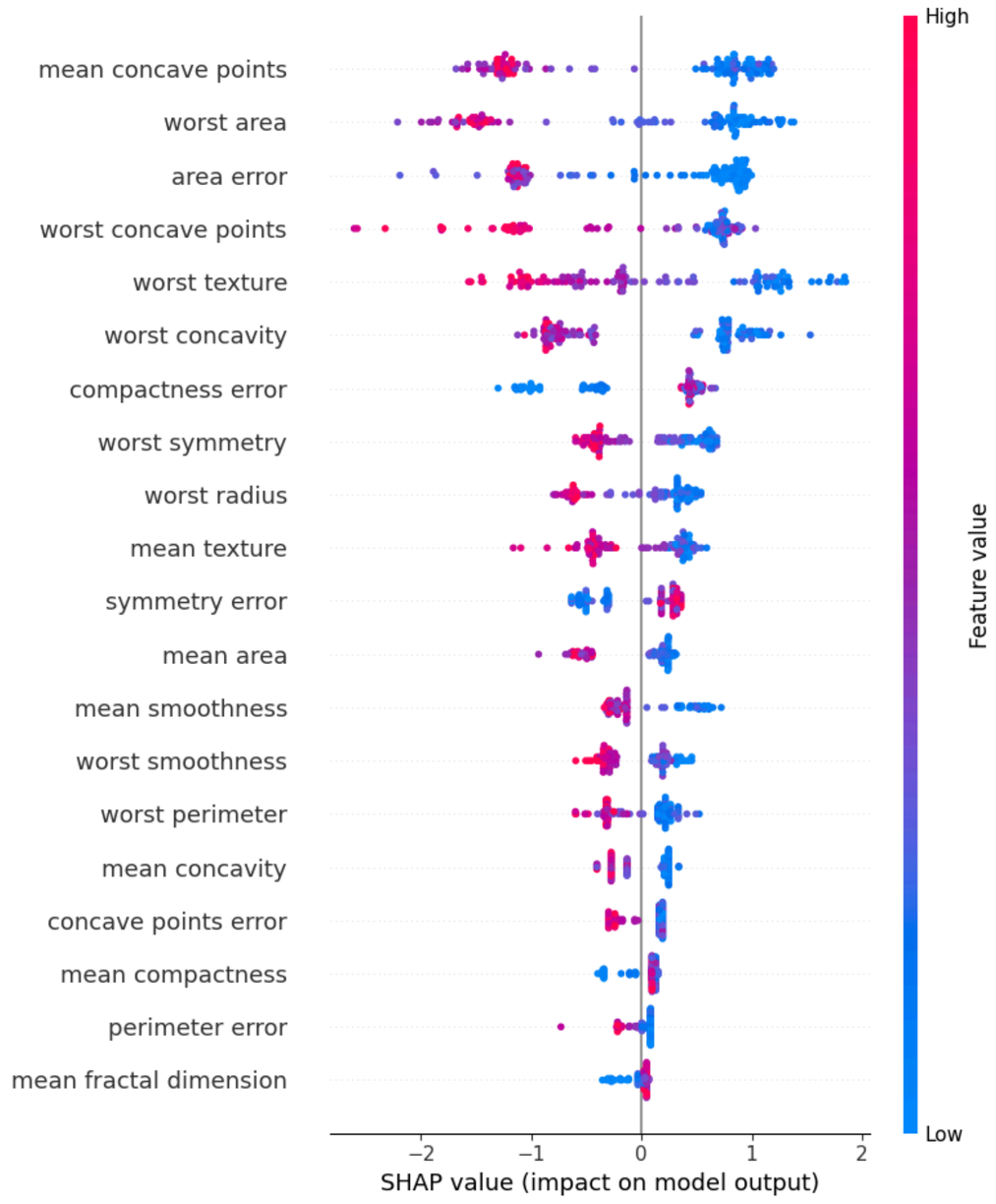
XGBClassifier

▶ Parameters

Shap value

```
explainer = shap.Explainer(model, X_train)
shap_values = explainer(X_test)

shap.summary_plot(shap_values, X_test)
```

Shap value



각 점들 : 각 feature의 각 샘플 값

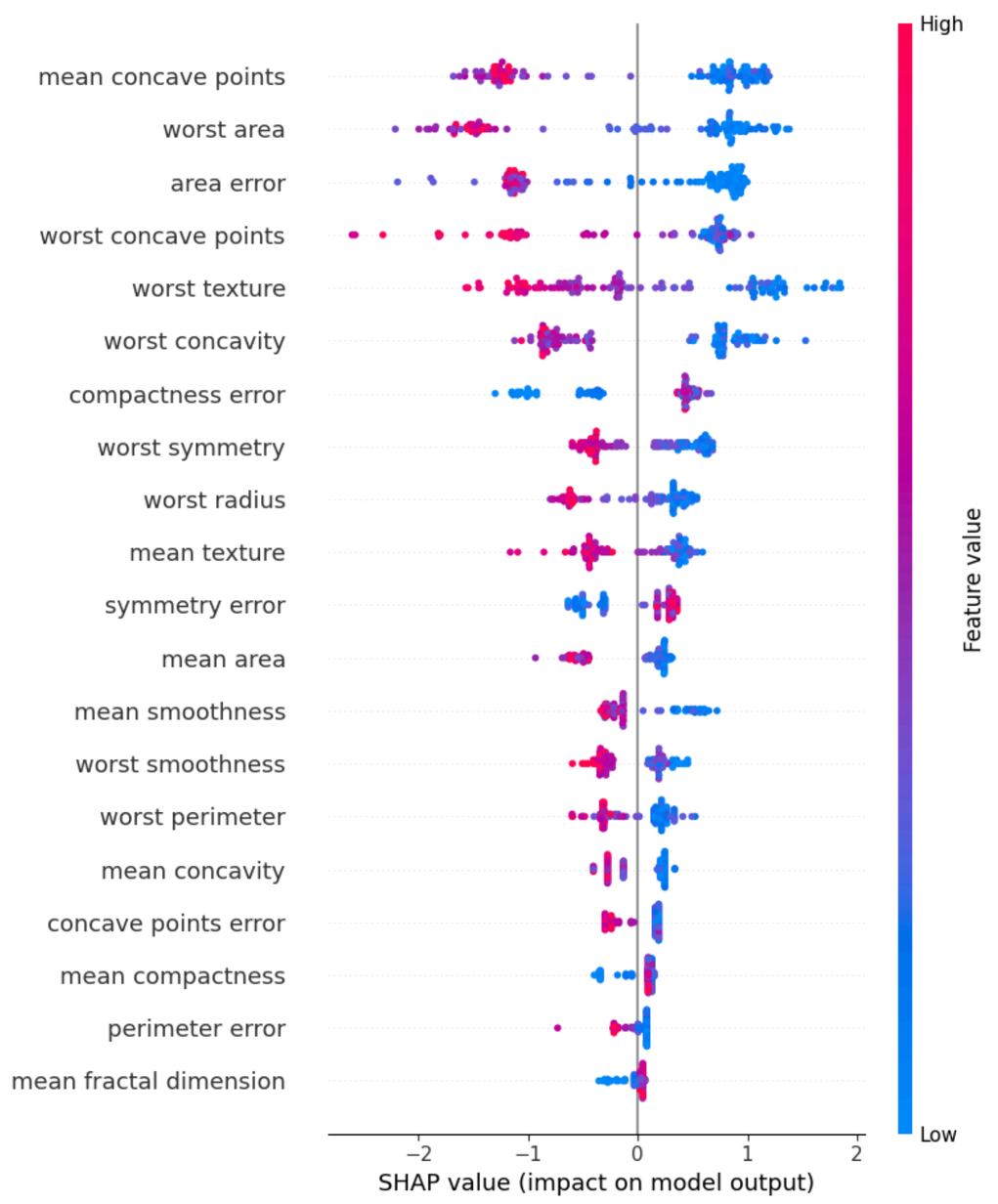
shap value : target값을 맞추는데 얼마나 영향을 줬는지

“mean_concave_points”

값이 작은 데이터들은 예측값이 큰 거에 영향을 많이 줌

값이 큰 데이터들은 예측값이 작은 거에 영향을 많이 줌

Shap value



좋은 feature

값이 골고루 퍼져있는 feature

shap value가 0에 모여있으면 제거 고려

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

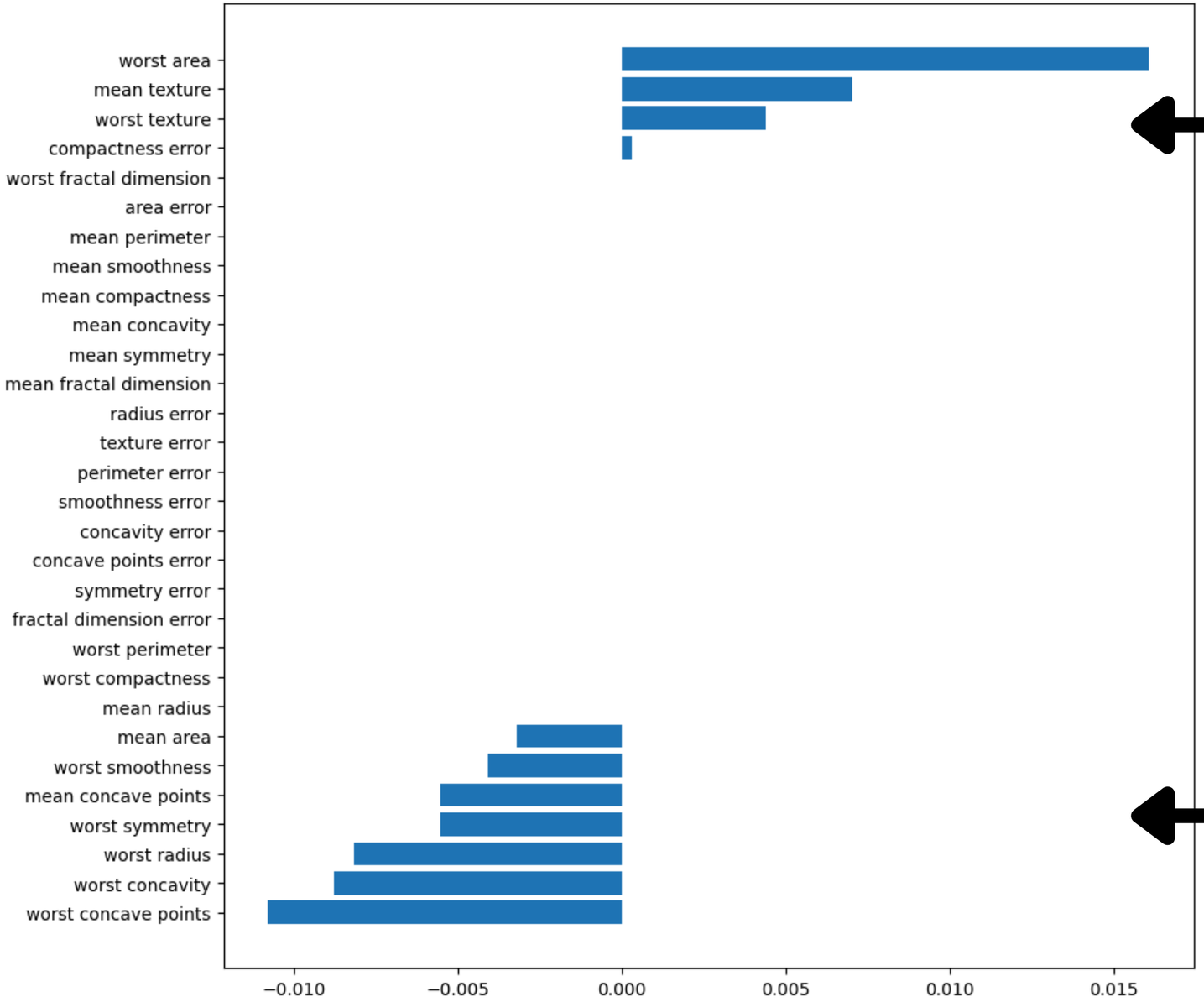
중요한 feature : Score_s = 0.12
덜 중요한 feature : Score_s = 0.82

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
```