고차원 데이터의 단점 : 모델이 과적합하는 경향이 있음

The curse of dimensionality

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Machine Learning Engineer, Faktion

datacamp

From observation to pattern

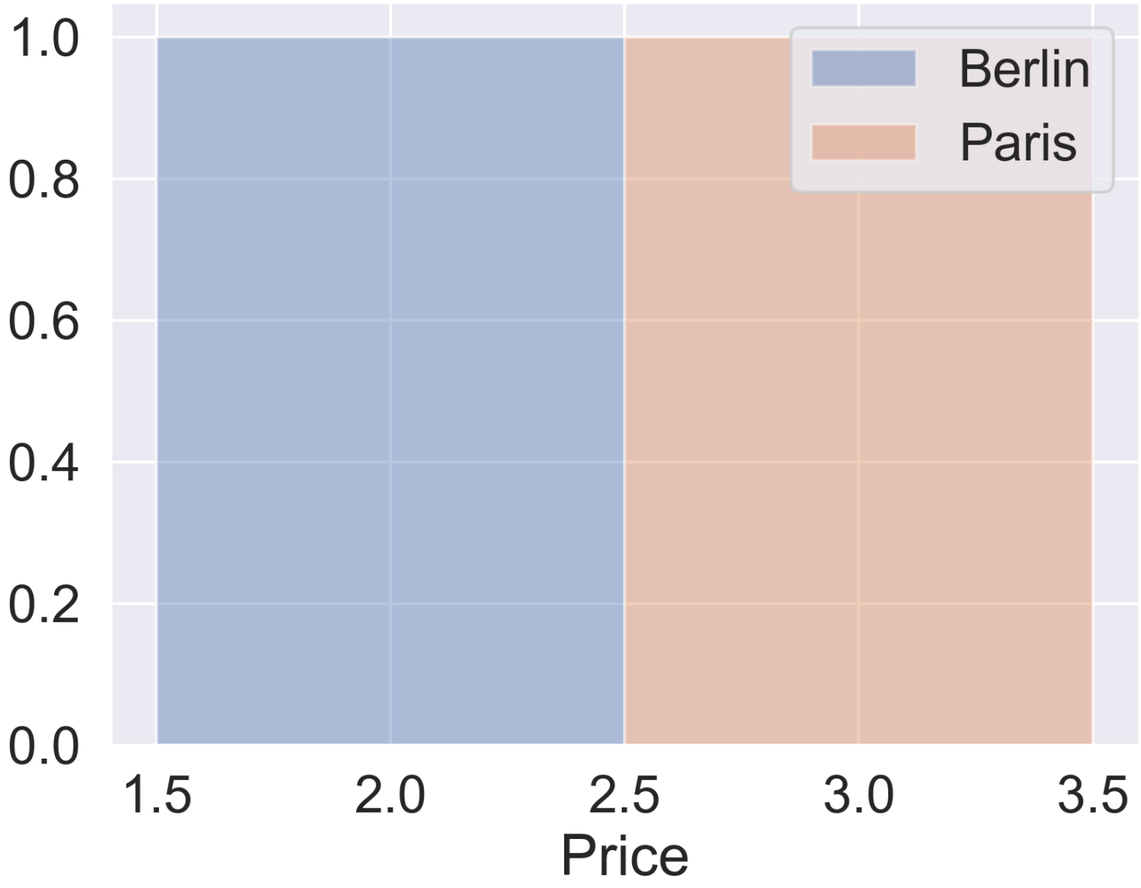
City	Price
Berlin	2
Paris	3

ex) 집의 몇 가지 feature을 바탕으로 집이 위치한 도시를 예측

From observation to pattern

City	Price
Berlin	2
Paris	3

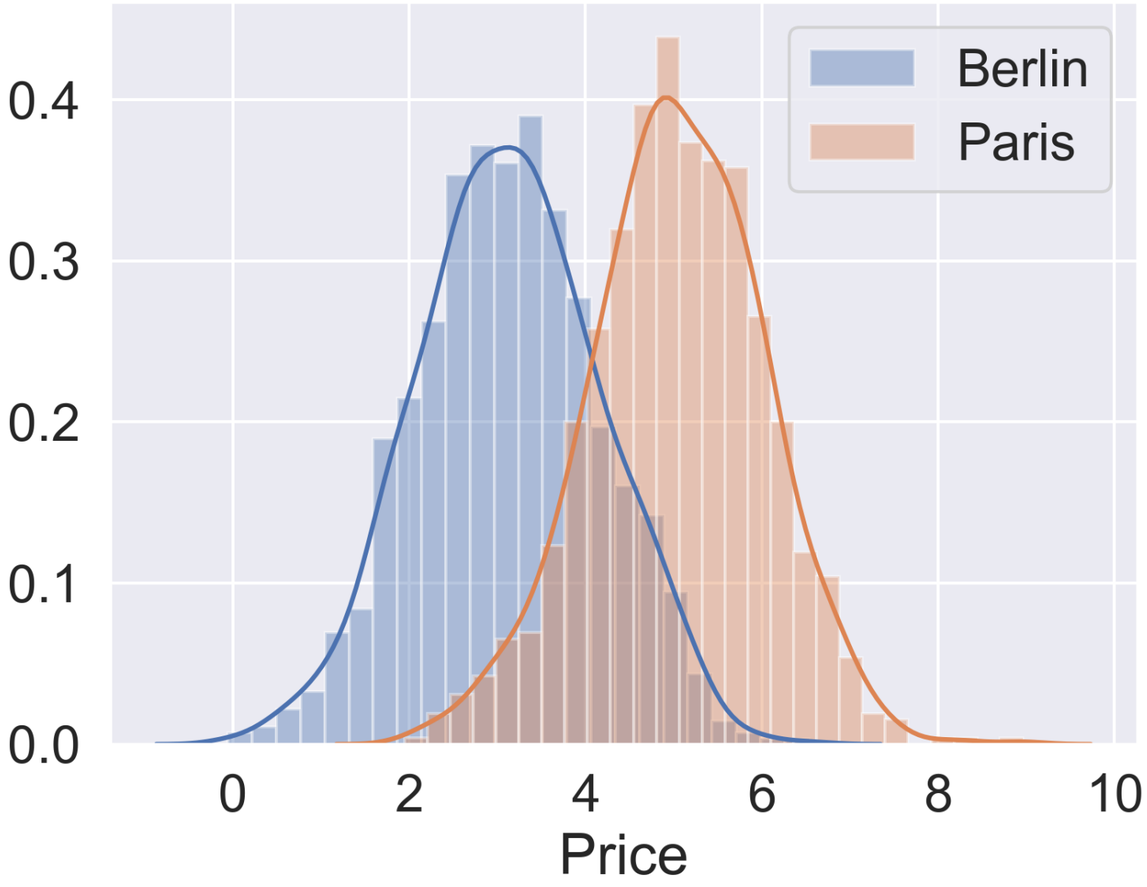
작은 data
-> model overfit될 것임



From observation to pattern

City	Price
Berlin	2.0
Berlin	3.1
Berlin	4.3
Paris	3.0
Paris	5.2
...	...

도시당 1000개의 관측치를 확장하면
가격 분포가 명확해지고, model은 과적합 없이
데이터를 학습할 수 있음



Building a city classifier - data split

Separate the feature we want to predict from the ones to train the model on.

```
y = house_df['City']

X = house_df.drop('City', axis=1)
```

Perform a 70% train and 30% test data split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Building a city classifier - model fit

Create a Support Vector Machine Classifier and fit to training data

```
from sklearn.svm import SVC

svc = SVC()

svc.fit(X_train, y_train)
```

Building a city classifier - predict

```
from sklearn.metrics import accuracy_score


print(accuracy_score(y_test, svc.predict(X_test)))
```

```
0.826
```

test set에서 82.6%의 정확도를 보임

```
print(accuracy_score(y_train, svc.predict(X_train)))
```

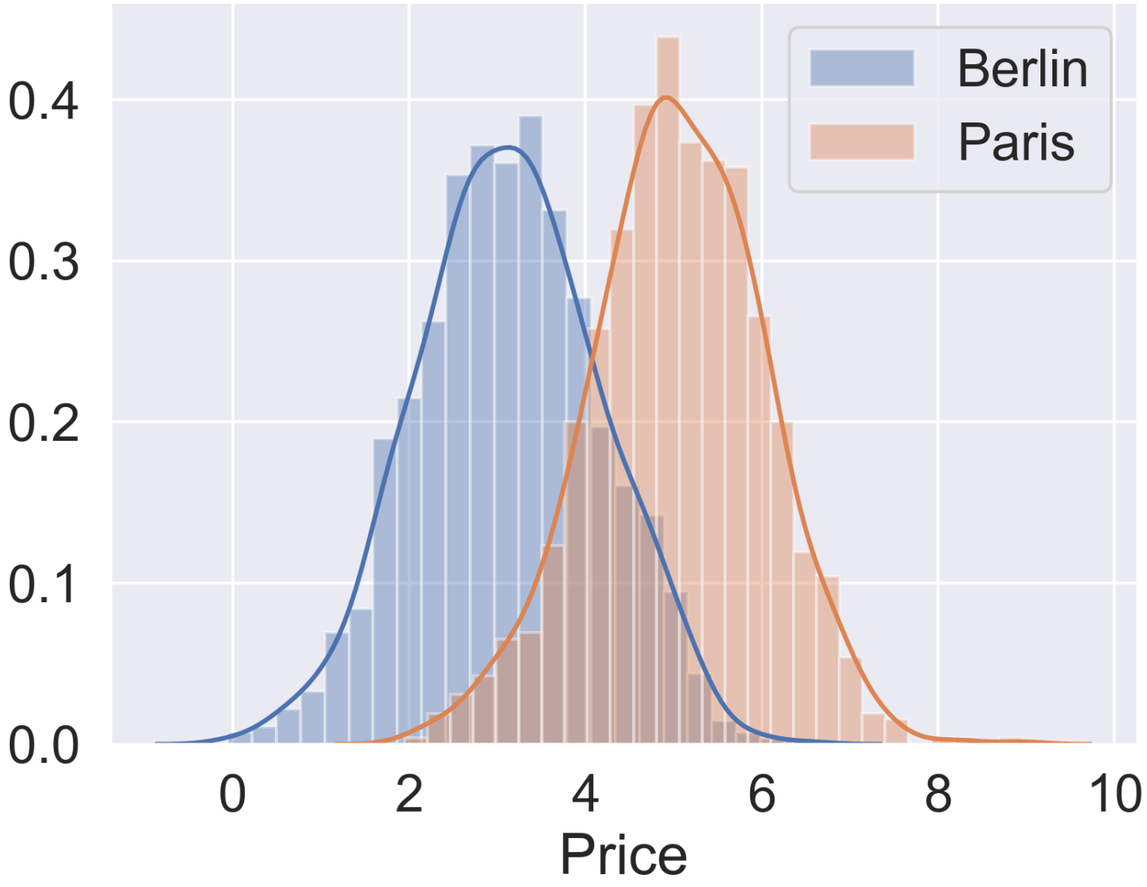
```
0.832
```

모델이 과적합했는지 알고 싶으면 train set의 accuracy를 살펴보면 됨
-> test set의 accuracy보다 훨씬 높으면 model이 잘 일반화되지 않지만 단순히 모든 훈련 예체를 암기했다는 결론을 내릴수 있지만
여기에는 그렇지 않음

Adding features

City	Price
Berlin	2.0
Berlin	3.1
Berlin	4.3
Paris	3.0
Paris	5.2
...	...

모델의 정확도를 높이려면 dataset에 feature을 추가해야 함



Adding features

City	Price	n_floors	n_bathroom	surface_m2
Berlin	2.0	1	1	190
Berlin	3.1	2	1	187
Berlin	4.3	2	2	240
Paris	3.0	2	1	170
Paris	5.2	2	2	290
...	...	...	...	...

추가하는 각 기능을 사용하여 dataset에서 주택의 관측 횟수를 늘려야 함.
그렇지 않으면 model이 쉽게 외울 수 있는 feature들의 독특한 조합이 많아질 것임.
실제로 과적합을 방지하려면 관측치의 수가 형상의 수에 따라 기하급수적으로 증가해야 함.
이는 고차원 dataset에서 문제가 되기 때문에 이 현상은 차원의 저주라고 알려져 있는데, 이를 해결할 방법은 차원 축소를 적용하는 것임.

Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

Features with missing values or little variance

DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Machine Learning Engineer, Faktion

이 강의에서는 분산이 충분하고 결측값이 너무 많지 않은 **feature**을 자동으로 선택함

datacamp

Creating a feature selector

```
print(ansur_df.shape)
```

```
(6068, 94)
```

```
from sklearn.feature_selection import VarianceThreshold


sel = VarianceThreshold(threshold=1)
sel.fit(ansur_df)


mask = sel.get_support()
print(mask)
```

분산이 낮은 기능은 서로 유하기 때문에 분석에 사용할 수 있는 정보가 거의 없음
-> 이를 제거하기 위해 VarianceThreshold()를 사용할 수 있음
최소 분산 임계값을 설정할 수 있는데, 여기서는 1로 설정하였음.
선택기를 dataset에 맞춘 후, 해당 get_support() 메서드는 각 feature의 분산이 임계값을 초과하는지 여부에 대해 True, False 값을 제공함 (boolean array a mask라고 함)

```
array([ True,  True, ..., False,  True])
```

Applying a feature selector

```
print(ansur_df.shape)
```

```
(6068, 94)
```

```
reduced_df = ansur_df.loc[:, mask]
print(reduced_df.shape)
```

```
(6068, 93)
```

feature 하나를 줄임

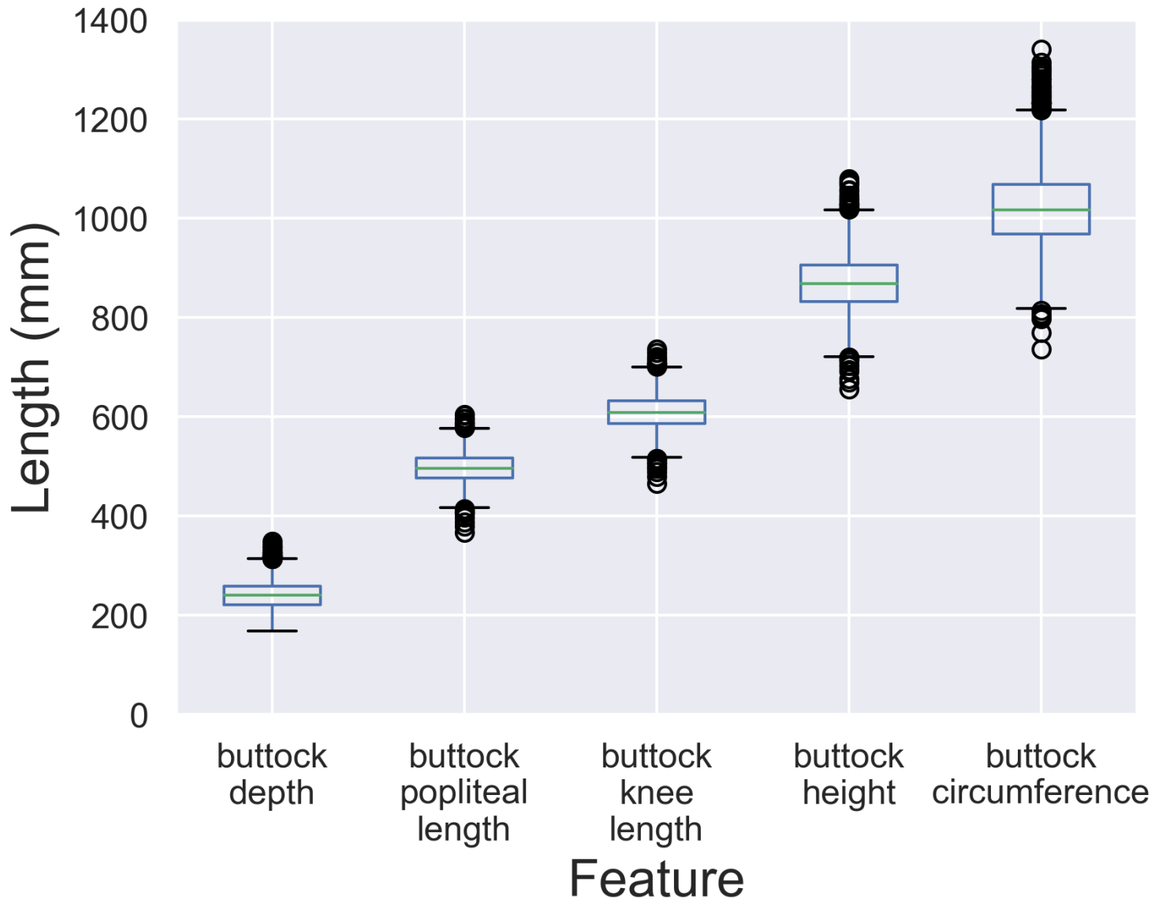
Variance selector caveats

```
buttock_df.boxplot()
```

분산 임계값의 한가지 문제 : 분산 값이 feature 간에 항상 해석하거나 비교하는 것이 쉽지 않음.

boxplot에서 서로 다른 측정값의 중위, 박스 녹색 수평선, 분산, 박스의 높이 및 수직선의 길이가 다르다는 것을 볼 수 있음.

이 dataset의 경우 값이 클수록 분산이 더 크므로 feature selection에 사용하기 전에 분산을 정규화해야함.



Normalizing the variance

```
from sklearn.feature_selection import VarianceThreshold


sel = VarianceThreshold(threshold=0.005)


sel.fit(ansur_df / ansur_df.mean())
mask = sel.get_support()
reduced_df = ansur_df.loc[:, mask]
print(reduced_df.shape)
```

selector를 fitting하기 전에 각 열을 해당 평균 값으로 나눔
정규화 후에는 dataset의 분산이 더 낮아짐.
=> 따라서 분산 임계값 0.005로 줄었음
selector를 dataset에 적용하면 feature 수가 45개로 절반 이상 감소함.

```
(6068, 45)
```

Missing value selector

Name	Type 1	Type 2	Total	HP	Attack	Defense
Bulbasaur	Grass	Poison	318	45	49	49
Ivysaur	Grass	Poison	405	60	62	63
Venusaur	Grass	Poison	525	80	82	83
Charmander	Fire	NaN	309	39	52	43
Charmeleon	Fire	NaN	405	58	64	58

결측값 삭제할 수도 있음

Missing value selector

Name	Type 1	Type 2	Total	HP	Attack	Defense
Bulbasaur	Grass	Poison	318	45	49	49
Ivysaur	Grass	Poison	405	60	62	63
Venusaur	Grass	Poison	525	80	82	83
Charmander	Fire	NaN	309	39	52	43
Charmeleon	Fire	NaN	405	58	64	58

Identifying missing values

```
pokemon_df.isna()
```

Name	Type 1	Type 2	Total	HP	Attack	Defense
False	False	False	False	False	False	False
False	False	False	False	False	False	False
False	False	False	False	False	False	False
False	False	True	False	False	False	False
False	False	True	False	False	False	False

Counting missing values

```
pokemon_df.isna().sum()
```

```
Name          0
Type 1        0
Type 2      386
Total         0
HP            0
Attack        0
Defense       0
dtype: int64
```

Counting missing values

```
pokemon_df.isna().sum() / len(pokemon_df)
```

숫자를 데이터 프레임의 총 행 수로 나누면 0과 1 사이의 결측값 비율을 얻게됨

```
Name        0.00
Type 1      0.00
Type 2      0.48
Total       0.00
HP          0.00
Attack      0.00
Defense     0.00
dtype: float64
```

Type 2는 절반 이상 결측값

Applying a missing value threshold

```
# Fewer than 30% missing values = True value
mask = pokemon_df.isna().sum() / len(pokemon_df) < 0.3
print(mask)
```

비율을 기반으로 특정 theshold보다 적은 feature에 대한 mask를 만들 수 있음 (여기서 0.3으로 설정)

```
Name          True
Type 1        True
Type 2       False
Total         True
HP            True
Attack        True
Defense       True
dtype: bool
```

Applying a missing value threshold

```
reduced_df = pokemon_df.loc[:, mask]


reduced_df.head()
```

Name	Type 1	Total	HP	Attack	Defense
Bulbasaur	Grass	318	45	49	49
Ivysaur	Grass	405	60	62	63
Venusaur	Grass	525	80	82	83
Charmander	Fire	309	39	52	43
Charmeleon	Fire	405	58	64	58

Let's practice

DIMENSIONALITY REDUCTION IN PYTHON

Pairwise correlation

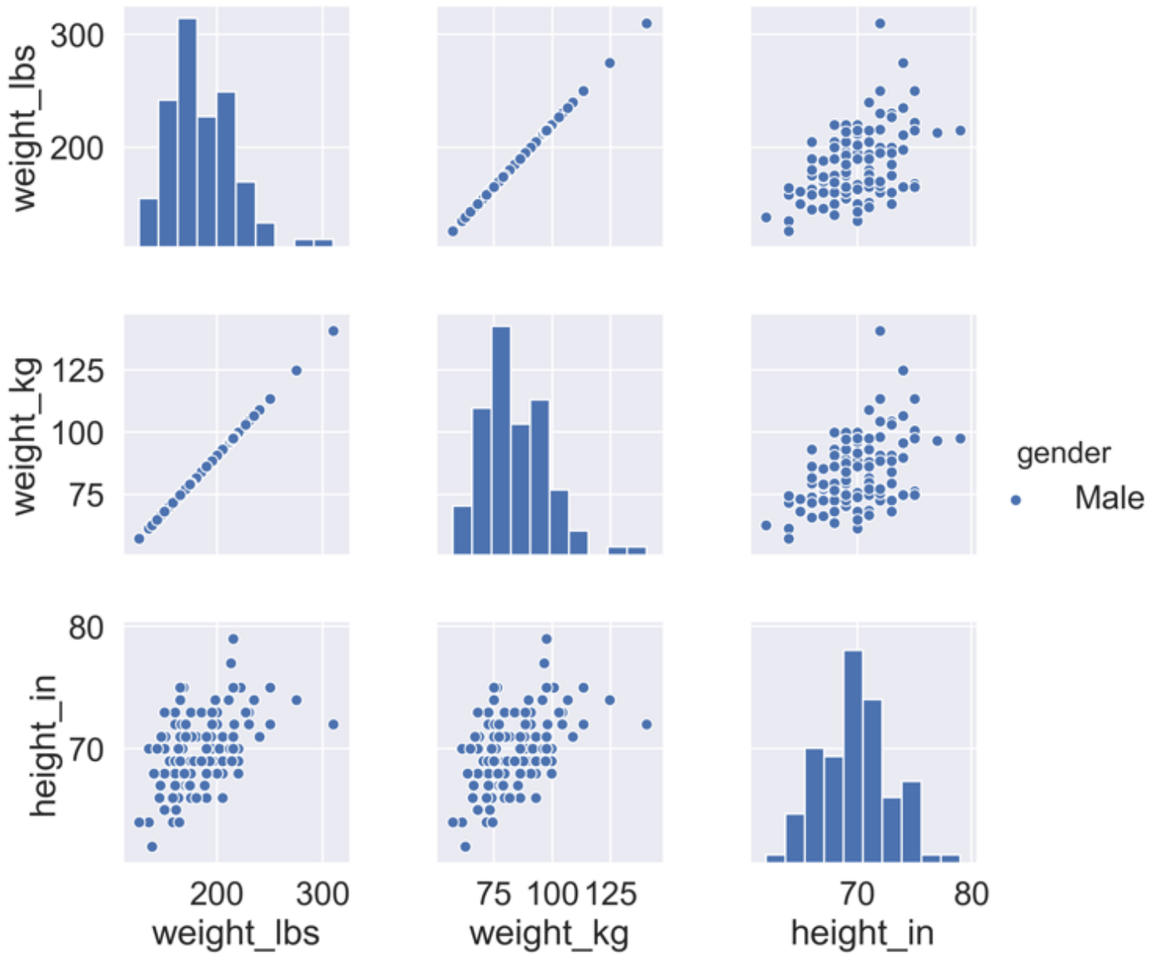
DIMENSIONALITY REDUCTION IN PYTHON

Jeroen Boeye
Machine Learning Engineer, Faktion

datacamp

Pairwise correlation

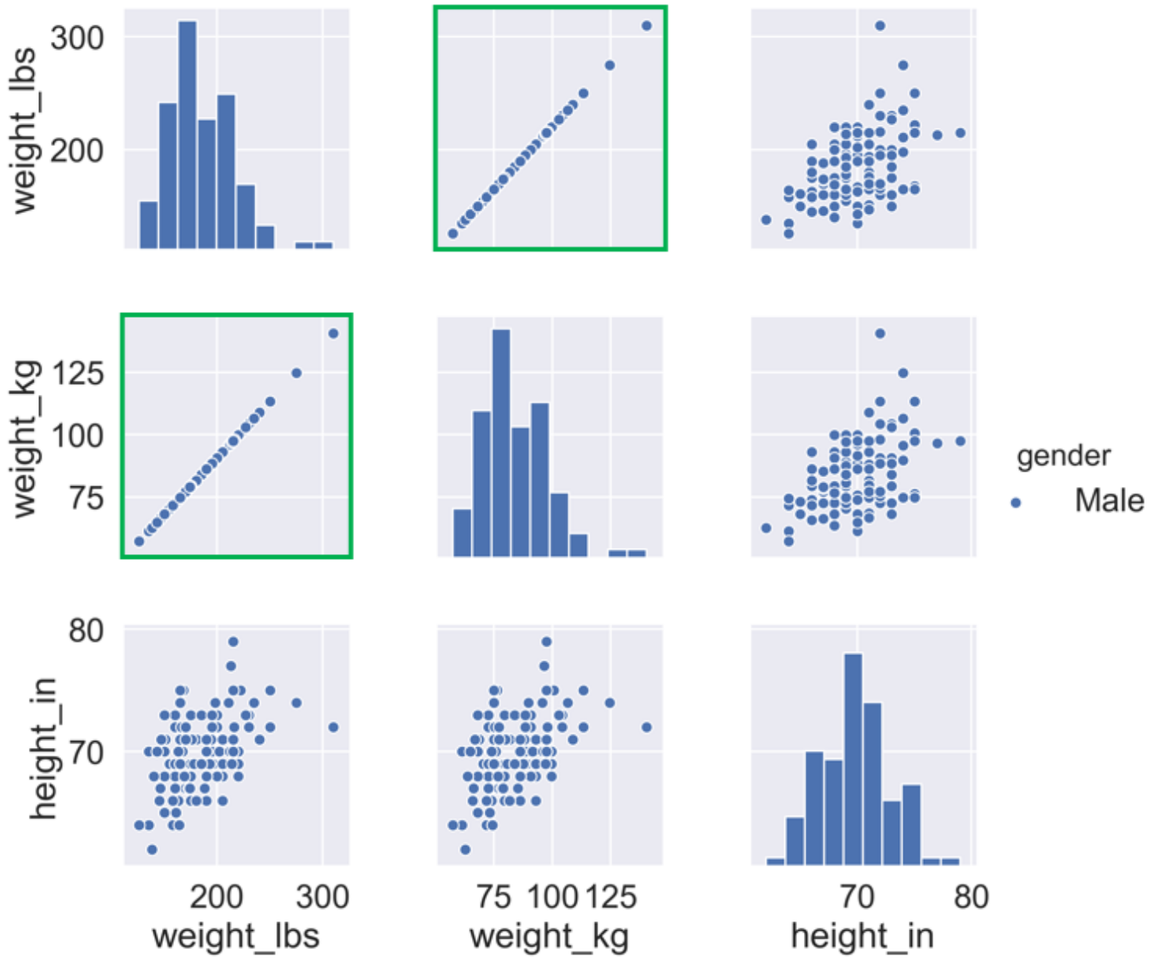
```
sns.pairplot(ansur, hue="gender")
```



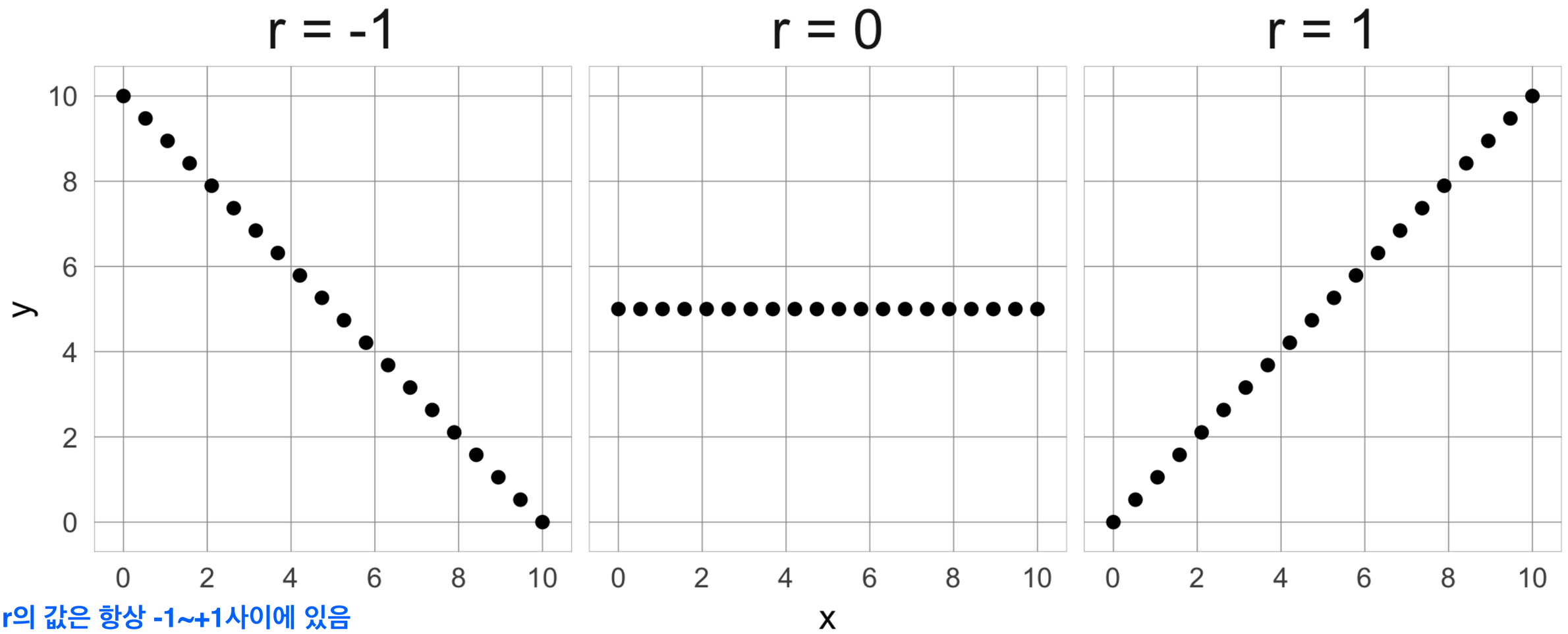
Pairwise correlation

```
sns.pairplot(ansur, hue="gender")
```

시각적으로 상관관계를 식별할 수 있음
-> 그러나 feature간의 상관관계를 정량화하기에는 부족함

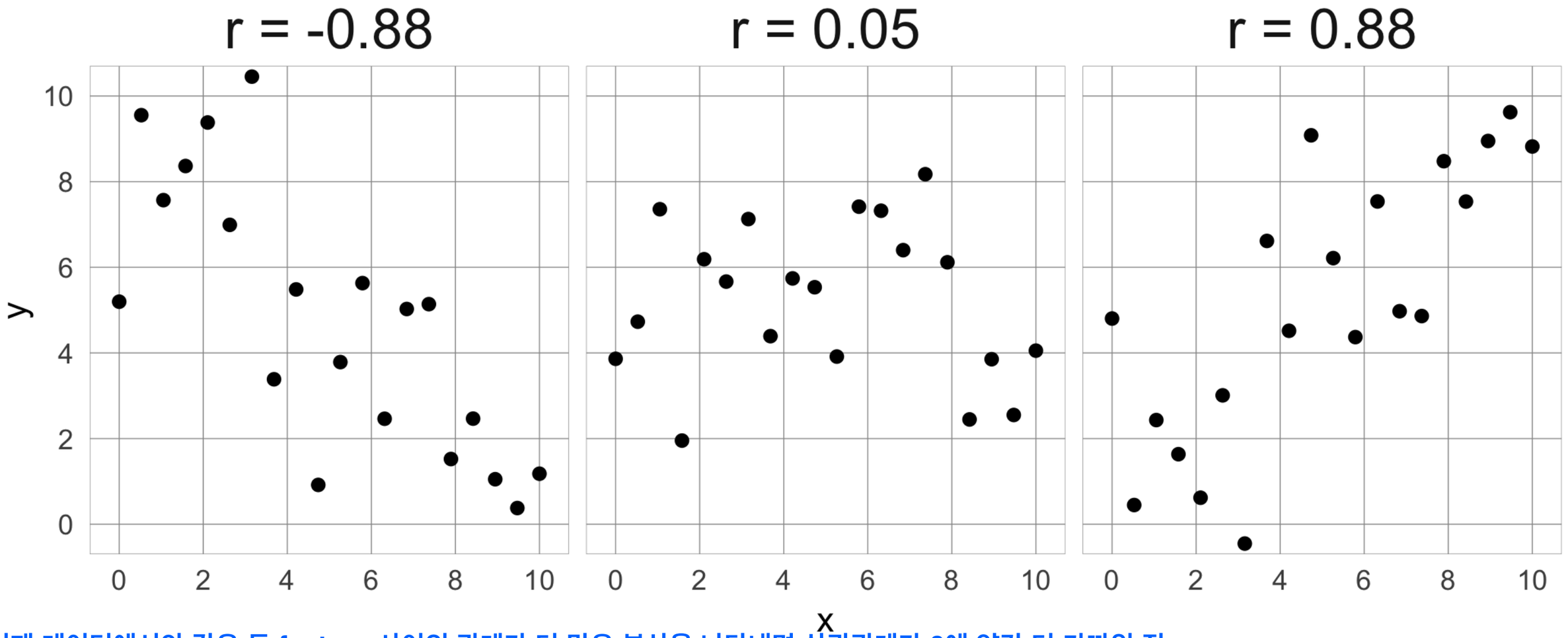


Correlation coefficient



r의 값은 항상 -1~+1사이에 있음
-1 : 음의 상관관계
+1 : 양의 상관관계

Correlation coefficient



실제 데이터에서와 같은 두 feature 사이의 관계가 더 많은 분산을 나타내면 상관관계가 0에 약간 더 가까워 짐.

Correlation matrix

```
weights_df.corr()
```

각 feature 간의 상관관계

	weight_lbs	weight_kg	height_in
weight_lbs	1.00	1.00	0.47
weight_kg	1.00	1.00	0.47
height_in	0.47	0.47	1.00

Correlation matrix

```
weights_df.corr()
```

A와 B의 상관관계는 B와 A의 상관관계와 같기 때문에 두번 표시

	weight_lbs	weight_kg	height_in
weight_lbs	1.00	1.00	0.47
weight_kg	1.00	1.00	0.47
height_in	0.47	0.47	1.00

Correlation matrix

```
weights_df.corr()
```

kg, lbs는 완벽하게 상관된 feature

Correlation matrix

```
weights_df.corr()
```

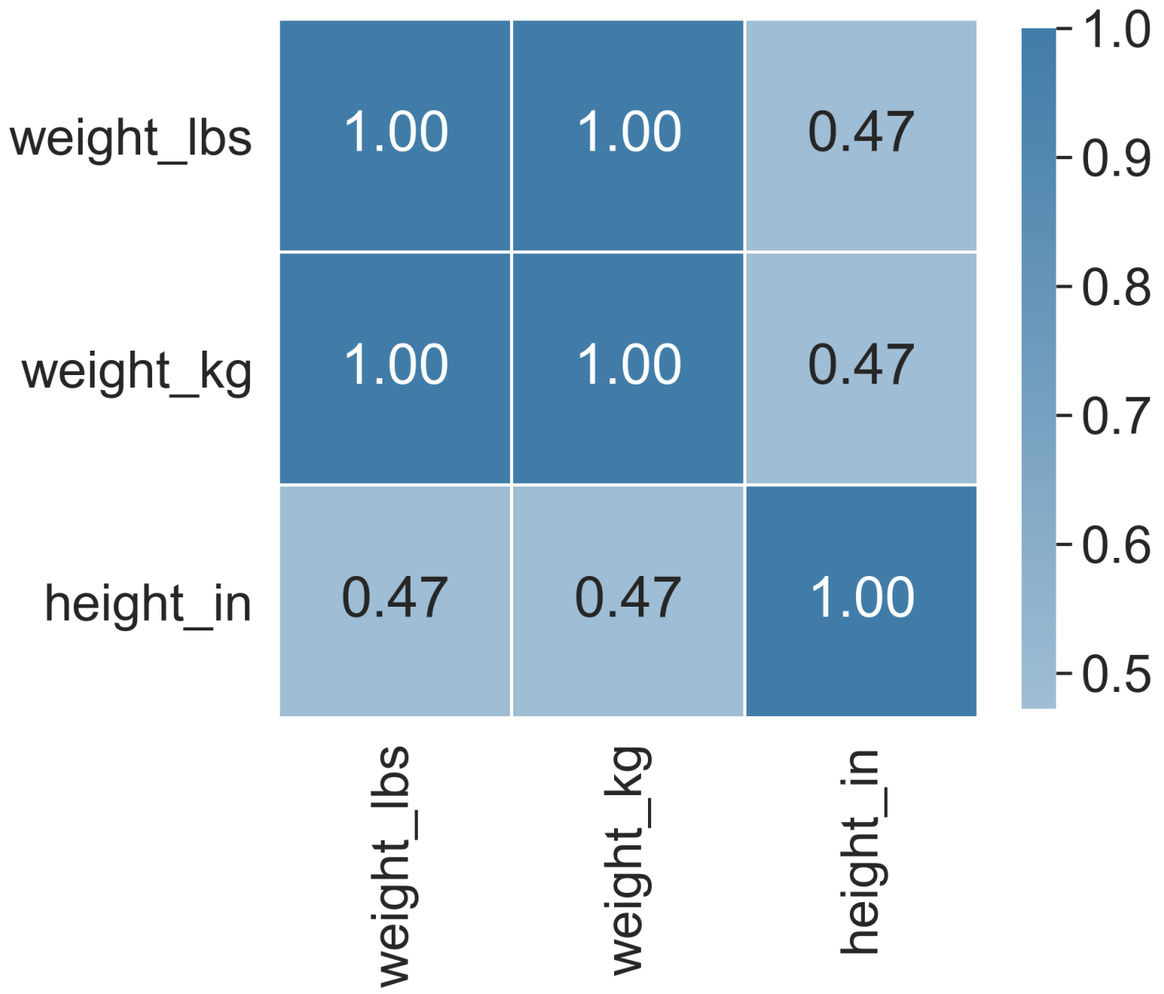
	weight_lbs	weight_kg	height_in
weight_lbs	1.00	1.00	0.47
weight_kg	1.00	1.00	0.47
height_in	0.47	0.47	1.00

Visualizing the correlation matrix

```
cmap = sns.diverging_palette(h_neg=10,
                             h_pos=240,
                             as_cmap=True)

sns.heatmap(weights_df.corr(), center=0,
            cmap=cmap, linewidths=1,
            annot=True, fmt=".2f")
```

사용자 지정 색상 팔레트와 스타일링 인수를 전달

heatmap을 사용하여 correlation matrix를 시각화 할 수 있음.



Visualizing the correlation matrix

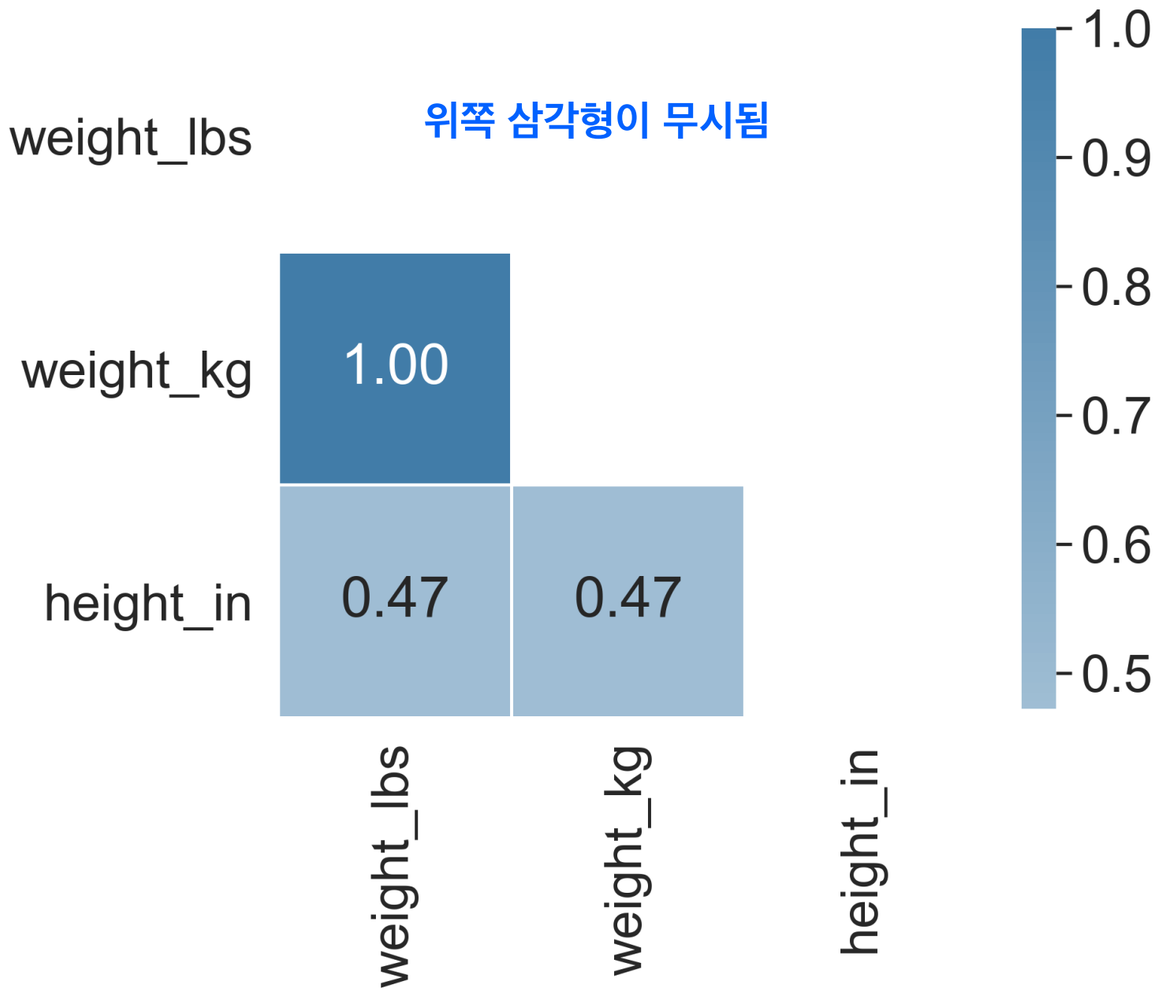
```
corr = weights_df.corr()


mask = np.triu(np.ones_like(corr, dtype=bool))
```

```
array([[ True,  True,  True],
       [False,  True,  True],
       [False, False,  True]])
```

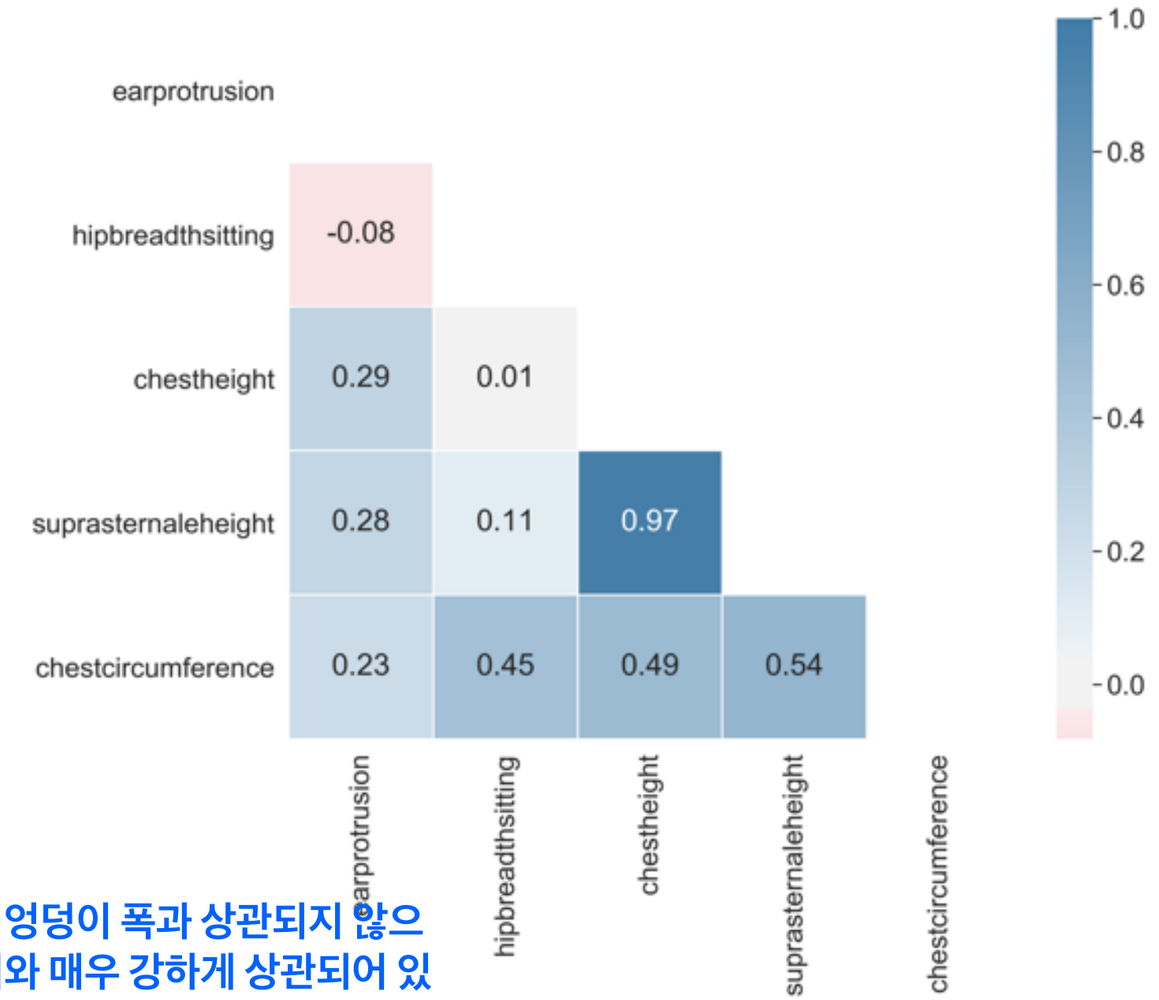
부울 마스크 생성한 후 one_like()함수를 사용하여 상관 행렬과 동일한 차원을 가진 True값으로 채워진 행렬을 만든 다음
이를 Numpy의 triu()로 전달하여, 삼각형 상단의 경우 모든 바상위 삼각셩 값을 False로 설정

Visualizing the correlation matrix

```
sns.heatmap(weights_df.corr(), mask=mask,
            center=0, cmap=cmap, linewidths=1,
            annot=True, fmt=".2f")
```



Visualising the correlation matrix



가슴 높이가 앉아 있는 동안 엉덩이 폭과 상관되지 않으며, 상완골 높이가 가슴 높이와 매우 강하게 상관되어 있음을 즉시 발견할 수 있음

Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON

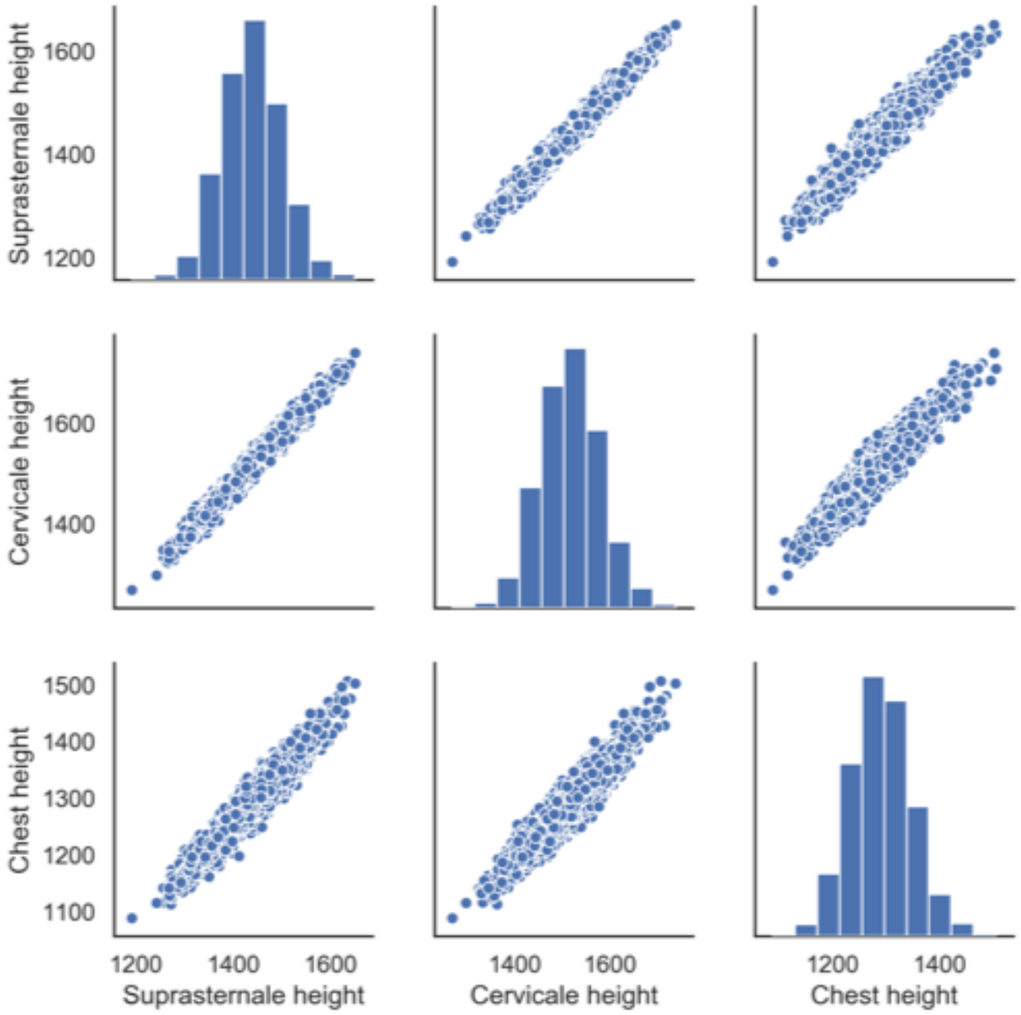
상관계수가 1 or -1인 경우 두 feature 중 하나를 삭제해야함.
또한, 상관계수가 -1 or +1에 가까운 feature은 동일하거나 유사한 feuatre의 측정값일 경우 삭제할 수 있음

Removing highly correlated features

DIMENSIONALITY REDUCTION IN PYTHON

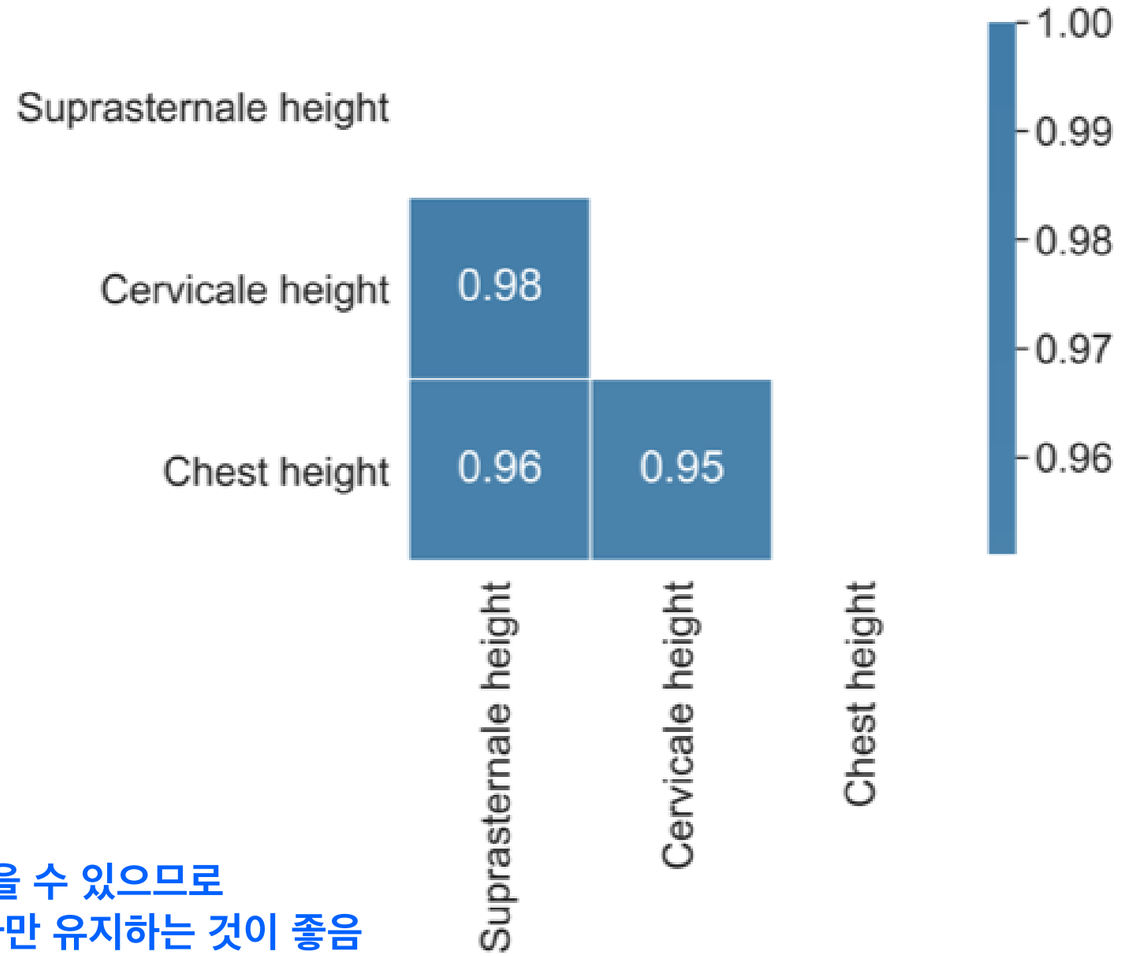
Jeroen Boeye
Machine Learning Engineer, Faktion

Highly correlated data



상완골과 경추는 가슴 부위에 있는 두 개의 뼈이기 때문에 이 세 가지 측정값은 항상 매우 유사

Highly correlated features



98%의 상관 계수를 얻을 수 있으므로
이러한 feature 중 하나만 유지하는 것이 좋음
단순성을 위해서뿐만 아니라 무의미한 차이들에 지나치게 맞추지 않도록 하기 위해서.

상관관계가 높은 feature을 삭제해도 너무 많은 정보가 손실되지 않을 것으로 확신하는 경우 임계값을 사용하여 해단 feature을 필터링할 수 있음

Removing highly correlated features

```
# Create positive correlation matrix
corr_df = chest_df.corr().abs()
# Create and apply mask
mask = np.triu(np.ones_like(corr_df, dtype=bool))
tri_df = corr_matrix.mask(mask)

tri_df
```

절대값을 취해 음의 상관 관계를 걸러냄

위쪽 삼각형을 위해 mask 생성

mask가 Ture 값을 갖는 dataframe의 모든 위치를 NA를 바꿈

	Suprasternale height	Cervicale height	Chest height
Suprasternale height	NaN	NaN	NaN
Cervicale height	0.983033	NaN	NaN
Chest height	0.956111	0.951101	NaN

Removing highly correlated features

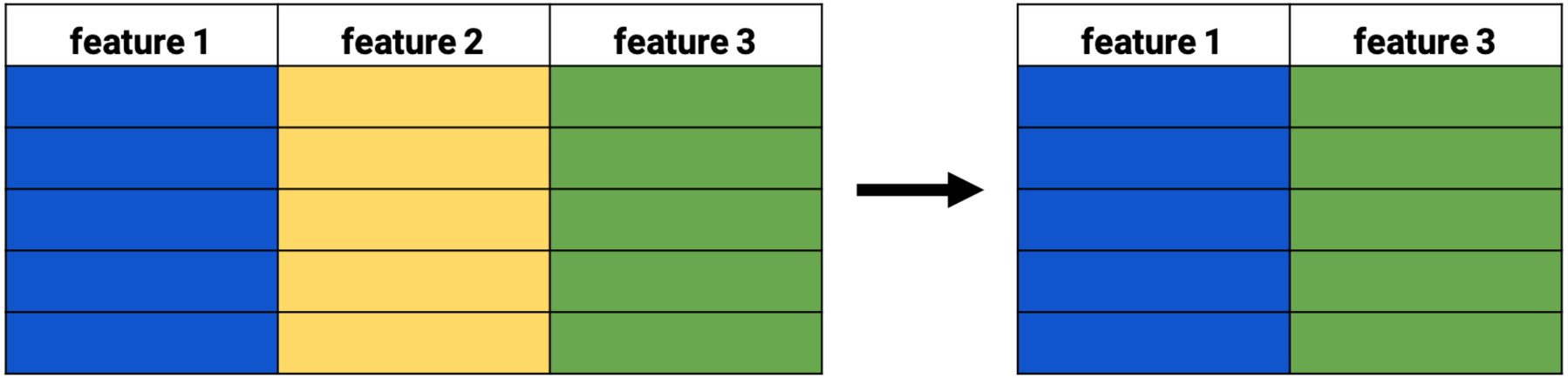
```
# Find columns that meet treshold
to_drop = [c for c in tri_df.columns if any(tri_df[c] > 0.95)]  임계값보다 강한 feature 걸러냄


print(to_drop)
```

```
['Suprasternale height', 'Cervicale height']
```

```
# Drop those columns
reduced_df = chest_df.drop(to_drop, axis=1)
```

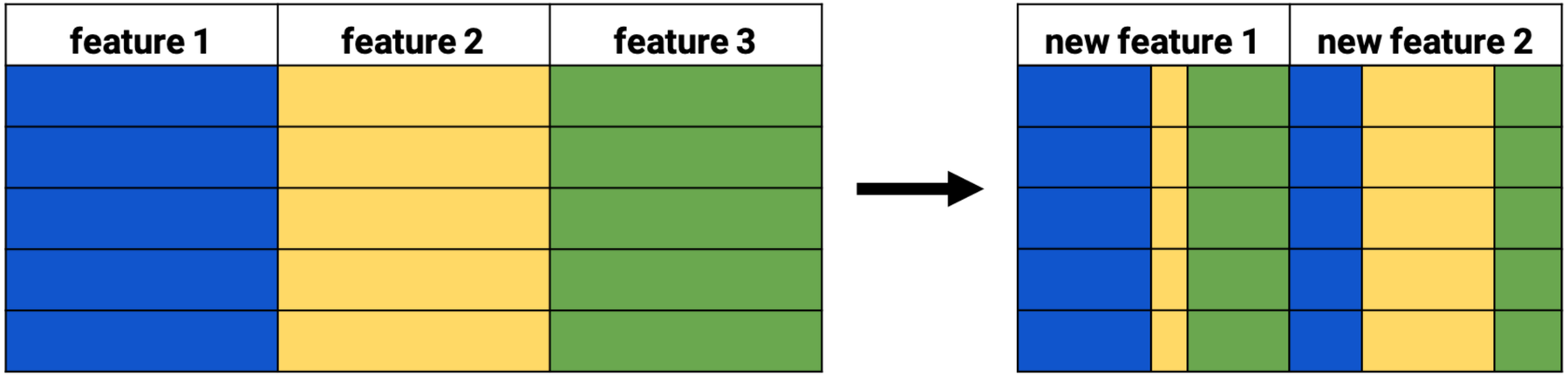
mask를 사용하여 행렬의 절반을 NA값으로 설정한 이유는 두 feature이 강한 상관관계를 가질 때 두 feature이 제거되지 않도록 하기 위함
drop을 사용하여 선택한 feature를 dataframe에서 삭제

Feature selection

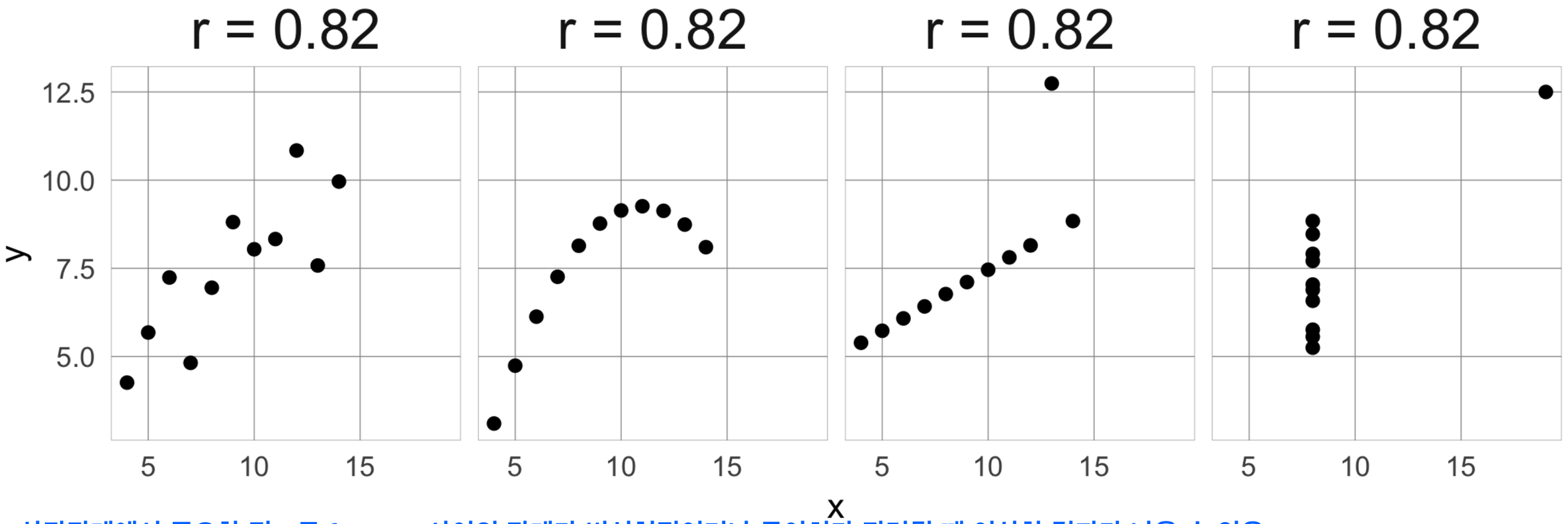


상관관계가 높은 feature를 제거하면 데이터에서 주용한 정보가 제거되는지 확실하지 않지만 차원을 줄여야 하는 경우에는 feature exdtraction을 고려

Feature extraction



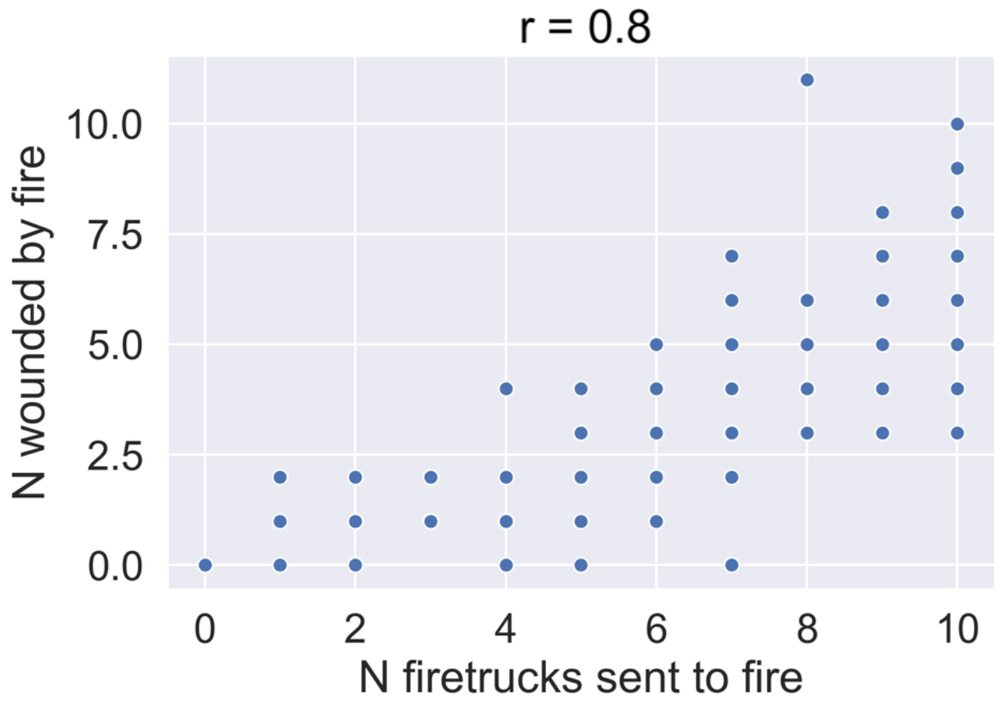
Correlation caveats - Anscombe's quartet



상관관계에서 중요한 점 : 두 feature 사이의 관계가 비선형적이거나 특이치가 관련될 때 이상한 결과가 나올 수 있음.
ex) 4가지 모두 동일한 상관 계수를 갖지만 시각적으로 다름

Correlation caveats - causation

```
sns.scatterplot(x="N firetrucks sent to fire",
                y="N wounded by fire",data=fire_df)
```



강한 상관관계는 인관관계를 의미하지 않음.
ex) 소방차 수는 해당 화재로 인해 부상당한 사람의 수와 상관 관계가 잇음.
더 많은 부상자들이 소방차를 더 많이 보내서 발생한다고 결론짓는 것은 잘된 것.

Let's practice!

DIMENSIONALITY REDUCTION IN PYTHON