



✓ 2. Multiple Linear Regression (다중 선형 회귀)

단순선형회귀와 같이 독립변수 x 의 변화에 따른 종속변수 y 의 변화를 선으로서 예측하는 기법인 데, 독립변수 x 가 여러개인 분석기법이다.

Simple
Linear Regression
단순 선형 회귀

$$y = mx + b$$



공부 시간

Multiple
Linear Regression
다중 선형 회귀

$$y = b + m_1x_1 + m_2x_2 + \dots + m_nx_n$$

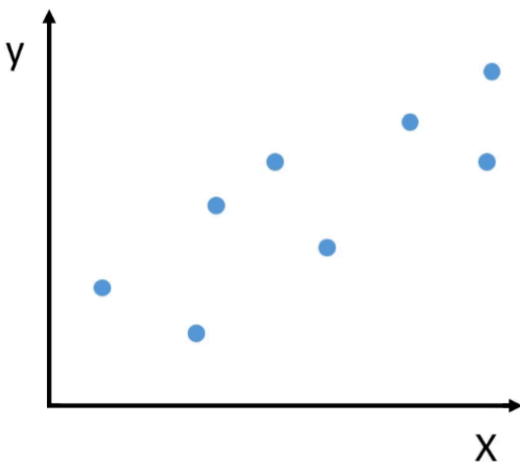


공부 시간

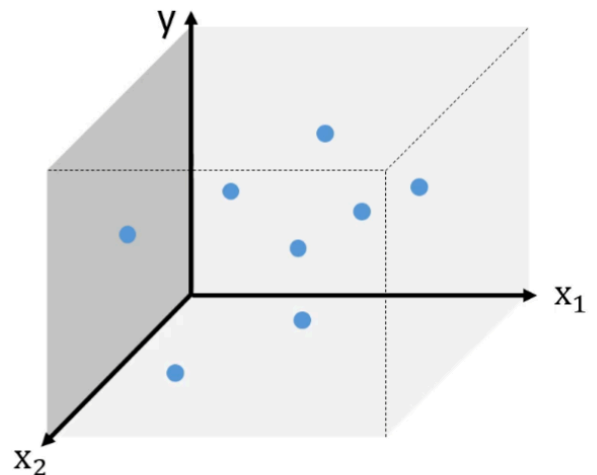


결석 횟수

독립변수 x 가 여러개 = 특성(feature)이 여러개라는 뜻이다. 하여 독립변수 2개를 이용해 선형회귀 모델을 만들면서 그 과정을 정리해보고, 중요한 용어나 내용들을 함께 정리해보겠다.



독립 변수 1개



독립 변수 2개

독립변수가 1개일 때는 이렇게 x, y 그래프의 2차원 형태로 표시할 수 있었는데 독립 변수가 2개가 되어 버리면 x_1, x_2 에 따라서 다른 값들이 3차원 공간에 표시 된다.

✓ 원-핫 인코딩

단어 집합의 크기를 벡터의 차원으로 하고, 표현하고 싶은 단어의 인덱스에 1의 값을 부여하고, 다른 인덱스에는 0을 부여하는 벡터 표현 방식입니다.

공부 장소	Home	Library	Cafe
Home	1	0	0
Library	0	1	0
Cafe	0	0	1

그림과 같이 표현 하는 것. 하지만 다중 공선성이란 문제가 생김 **다중 공선성(Multicollinearity)**이란 독립변수들 간에 서로 강한 상관 관계를 가지면서 회귀계수 추정의 오류가 나타나는 문제, 즉 하나의 독립변수가 다른 독립변수에 영향을 줄수 있다는 겁니다.

$$D1 + D2 + D3 = 1$$

$$D3 = 1 - (D1 + D2)$$

	D1	D2	D3
공부 장소	Home	Library	Cafe
Home	1	0	0
Library	0	1	0
Cafe	0	0	1

이렇게 높은 상관 관계를 가지면 문제가 발생할 수 있으므로 해결하는 방법은 아래와 같이 더미 데이터를 하나 삭제하는 방법입니다.

다중공선성

※ Dummy Column 이 n 개면? n-1 개만 사용


공부 장소	Home	Library	Cafe
Home	1	0	0
Library	0	1	0
Cafe	0	0	1

삭제

위와 같이 더미컬럼에서 한개를 삭제하고 모델을 만들어 주면 됩니다. n 개이면 $n-1$ 개를 사용하면 됩니다. 이 문제를 **Dummy variable trap** 이라고도 표현 합니다. 우리 데이터에서는 카페를 삭제하고 사용하는 것입니다.


```
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

 Mounted at /content/drive

```
dataset = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/활용편7-머신러닝/ScikitLearn/Multipl
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

X

 array([[0.5, 3, 'Home'],
[1.2, 4, 'Library'],
[1.8, 2, 'Cafe'],
[2.4, 0, 'Cafe'],
[2.6, 2, 'Home'],
[3.2, 0, 'Home'],
[3.9, 0, 'Library'],
[4.4, 0, 'Library'],
[4.5, 5, 'Home'],
[5.0, 1, 'Cafe'],
[5.3, 2, 'Cafe'],
[5.8, 0, 'Cafe'],
[6.0, 3, 'Library'],
[6.1, 1, 'Cafe'],
[6.2, 1, 'Library'],
[6.9, 4, 'Home'],
[7.2, 2, 'Cafe'],
[8.4, 1, 'Home'],
[8.6, 1, 'Library'],
[10.0, 0, 'Library']], dtype=object)

```
from sklearn.compose import ColumnTransformer # Column Transformer은 여러 transformer 을 column에
from sklearn.preprocessing import OneHotEncoder # 원핫 인코딩을 위한 클래스
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(drop='first'), [2])], remainder='pas
```

첫번째 - 'encoder' : 원핫 인코딩을 적용하기 위해 사용

두번째 - OneHotEncoder : 첫번째 인코딩을 수행할 클래스(원핫인코더) 객체를 넣어준다,

(drop='first'), [2] : 다중 공정성 문제를 해결하기 위해 첫번째 컬럼을 삭제하고 2개만 사용하겠다.

remainder='passthrough' : 원하는 코딩을 적용하지 않는 데이터들은 그냥 그대로 둔다.

```
# 원하는 값을 인코딩 후에 다시 X 에 입력한다.
```

```
X = ct.fit_transform(X)
```

X

```
# 1 0 : Home
```

```
# 0 1 : Library
```

```
# 0 0 : Cafe
```

```

→ array([[1.0, 0.0, 0.5, 3],
        [0.0, 1.0, 1.2, 4],
        [0.0, 0.0, 1.8, 2],
        [0.0, 0.0, 2.4, 0],
        [1.0, 0.0, 2.6, 2],
        [1.0, 0.0, 3.2, 0],
        [0.0, 1.0, 3.9, 0],
        [0.0, 1.0, 4.4, 0],
        [1.0, 0.0, 4.5, 5],
        [0.0, 0.0, 5.0, 1],
        [0.0, 0.0, 5.3, 2],
        [0.0, 0.0, 5.8, 0],
        [0.0, 1.0, 6.0, 3],
        [0.0, 0.0, 6.1, 1],
        [0.0, 1.0, 6.2, 1],
        [1.0, 0.0, 6.9, 4],
        [0.0, 0.0, 7.2, 2],
        [1.0, 0.0, 8.4, 1],
        [0.0, 1.0, 8.6, 1],
        [0.0, 1.0, 10.0, 0]], dtype=object)

```

✓ 데이터 세트 분리

```

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0) # 20 %

```

✓ 학습 (다중 선형 회귀)

```

from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)

```

```

→ ▼ LinearRegression
   LinearRegression()

```

✓ 예측 값과 실제 값 비교 (테스트 세트)

```

y_pred = reg.predict(X_test)
y_pred # 예측 값

```

```

→ array([ 92.15457859,  10.23753043, 108.36245302,  38.14675204])

```

```

y_test # 실제 값

```

```

→ array([ 90,    8, 100,  38])

```

```
reg.coef_ # 독립변수 (공부장소(집), 공부장소(도서관), 공부시간, 결석)
# 마이너스면 나쁜 결과 즉 집에서 공부하는게 가장 나쁜 결과 카페는 0이므로
# 공부시간은 1시간씩 증가할때마다 10.4점씩 올라가고, 결석은 1번 할때마다
# 1.64점씩 나쁜 영향을 준다.
```

```
⇒ array([-5.82712824, -1.04450647, 10.40419528, -1.64200104])
```

```
reg.intercept_ # y 절편
```

```
⇒ 5.365006706544776
```

✓ 모델 평가

```
reg.score(X_train, y_train) # 훈련 세트 - 96.2점
```

```
⇒ 0.9623352565265527
```

```
reg.score(X_test, y_test) # 테스트 세트 - 98.59점
```

```
⇒ 0.9859956178877446
```

✓ 다양한 평가 지표 (회귀 모델)

1. MAE (Mean Absolute Error)(평균절대오차) : 실제 정답 값과 예측 값의 차이를 절댓값으로 변환한 뒤 합산하여 평균을 구한다. 특이값이 많은 경우에 주로 사용된다. 값이 낮을수록 좋다.

$$MAE = \frac{\sum |y - \hat{y}|}{n}$$

2. MSE (Mean Squared Error)(평균 제곱 오차) : 실제 정답 값과 예측 값의 차이를 제곱한 뒤 평균을 구한다. 값이 낮을수록 좋다.

$$MSE = \frac{\sum_{i=1}^n (y - \hat{y})^2}{n}$$

3. RMSE (Root Mean Squared Error)(평균 제곱근 오차) : MSE에 루트는 씌워서 에러를 제공해서 생기는 값의 왜곡이 줄어든다. 값이 낮을수록 좋다.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y - \hat{y})^2}{n}}$$

4. R2 : 결정 계수. 데이터의 분산을 기반으로 한 평가 지표. 1에 가까울 수록 좋다.

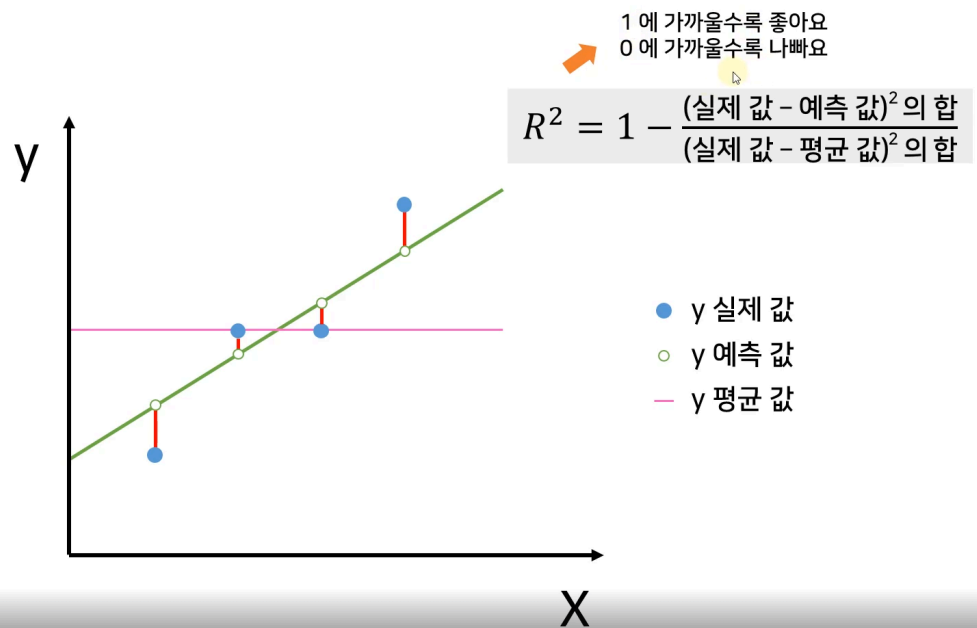
$$R^2 = 1 - \frac{SSE}{SST}$$

$$SSE = \sum_{i=1}^n (y^i - \hat{y}^i)^2 \Rightarrow (\text{실제값} - \text{예측값})^2 \text{의 합}$$

$$SST = \sum_{i=1}^n (y^i - \bar{y})^2 \Rightarrow (\text{실제값} - \text{평균값})^2 \text{의 합}$$

- \hat{y} : y 의 예측값
- \bar{y} : y 의 평균 값

Evaluation



R2 는 1에 가까울수록, 나머지는 0에 가까울수록 좋음

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, y_pred) # 실제 값, 예측 값 # MAE
```

⇒ 3.2253285188287997

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred) # MSE
```

⇒ 19.900226981514916

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred, squared=False) # RMSE. squared=False -> 제곱하지 말라는 뜻
```

⇒ 4.460967045553566

```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred) # R2 . 선형회귀 모델평가가 이 R2 스코어 로 계산 된다.
```

⇒ 0.9859956178877446

