



# PART II-03

## 머신러닝 기본

# 시간계획

- 오늘도 파이팅 입니다.^^

시간	학습내용
09:00~10:00	회귀모델 이해 및 성능 최적화를 위한 feature engineering 이해
10:20~11:20	LinearRegression(단순선형회귀) 실습
11:40~12:40	(회귀모델 데이터 학습 및 예측
12:40~14:00	즐거운 점심 시간
14:00~15:00	앙상블 이해 및 알고리즘 살펴보기
15:20~16:20	랜덤포레스트 실습
16:40~17:50	XGBoost 실습



## 학습내용

- 머신러닝 회귀 학습 모델을 이해하고 만든다.
- 회귀 모델의 평가 방법 MSE, RMSE, MAE를 안다.
- K-Fold 교차검증을 알고, 학습모델에 활용한다.
- Feature Engineering 이해하고, 방법을 안다.

---

# 지도 학습(Supervised Learning) 회귀(Regression)

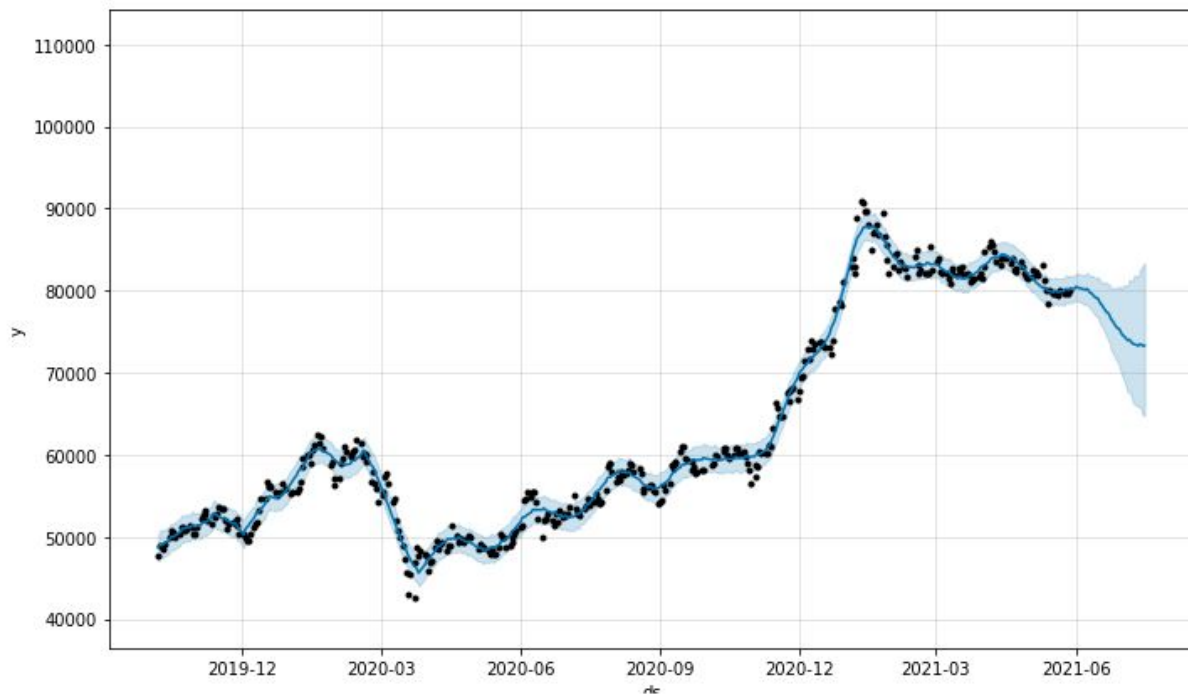
# 머신러닝 Estimator(학습기) 선택

- 코로나19 백신 2차까지 접종, 전국민 70%까지 접종이 완료 되었다.  
With 코로나 상황, 이제 마음껏 여행을 다닐 수 있다. 오늘 금요일, 내일 가족 또는 애인과 여행을 가기로 했다.
- 내일 비가 올까 안 올까? 기온은 몇 도가 될까?



# 회귀 모델의 예

- S전자 주가 예측



---

# 지도 학습(Supervised Learning) 회귀 모델(Regression)

# 회귀 분석 개념

- 데이터 변수들 간의 함수 관계를 파악하여 통계적 추론을 하는 기술
- 독립변수(features)에 대한 종속 변수 값의 평균을 구하는 방법

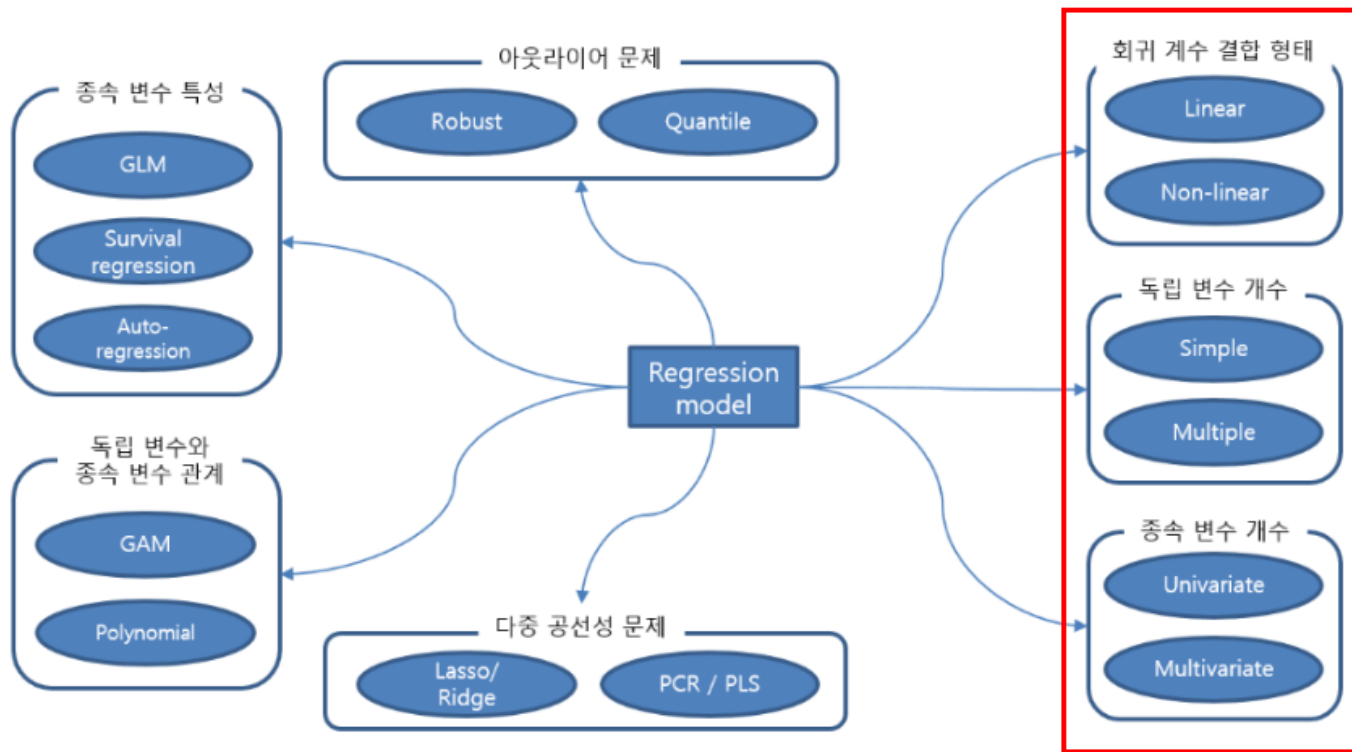
$$y = h(x_1, x_2, x_3, \dots, x_k; \beta_1, \beta_2, \beta_3, \dots, \beta_k) + \epsilon$$

- 조건에 따른 평균을 구하는 함수  $h(x) \Rightarrow$  회귀 모델, 가설 함수

All models are wrong but some are useful - 조지박스(통계학자)



# 회귀 분석의 종류



<https://bangu4.tistory.com/100>

ML의 회귀모델  
분류 기준

# 회귀 분석의 종류

- 선형(Linear) vs 비선형(Non-Linear)
  - 선형과 비선형을 결정하는 대상은 **회귀계수**에 따른 분류
  - **독립변수 개수와는 무관함.**
- 단순(Simple) vs 다중(Multiple)
  - 종속변수  $y$ 를 구하기 위한 **독립변수( $x$ )의 개수**에 따른 분류
  - 단순(단항) 선형 회귀(Simple Linear Regression) : 독립변수 1개
  - 다중(다항) 선형 회귀(Multiple Linear Regression) : 독립변수 2개 이상
- Univariate(단변량) vs Multivariate(다변량)
  - 구하고자 하는  **$y$ (종속변수)의 개수**에 따른 분류
  - 단변량 회귀 모델(Univariate Linear Regression) : 종속변수 1개
  - 다변량 회귀 모델(Multivariate Linear Regression) : 종속변수 2개 이상

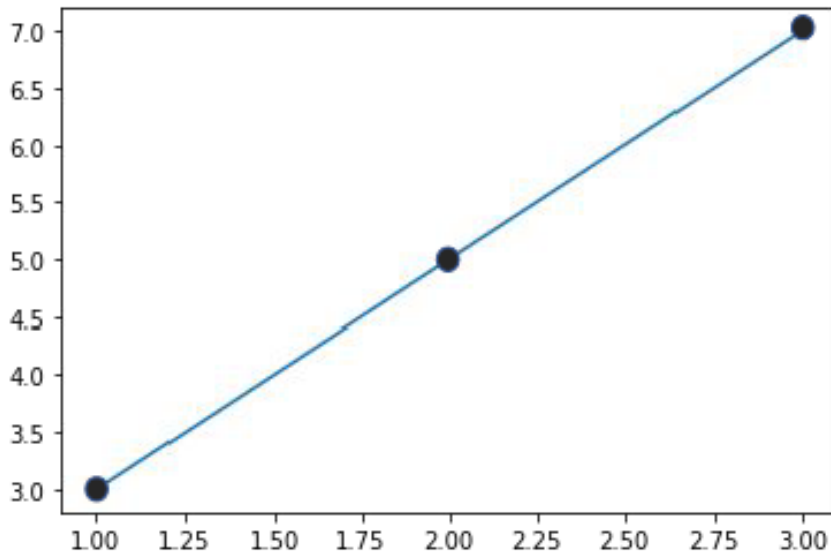
# 머신러닝-가설함수

W : Weight(가중치)=회귀계수(coefficient)

b : y 절편 = 편향(offset) = intercept

$$\hat{y} = H(x) = W * x + b$$

X	Y
1	3
2	5
3	7

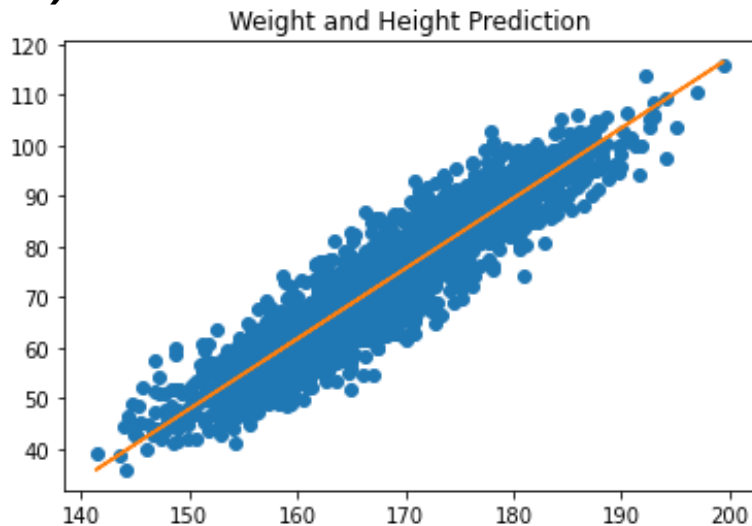
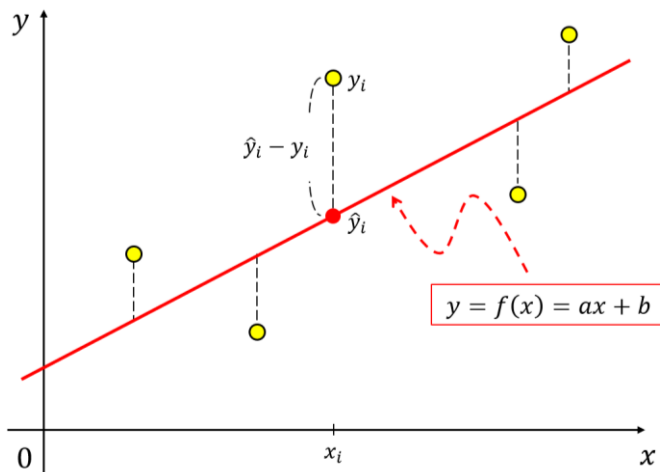


# Linear Regression(선형회귀)

- 선형함수를 이용해서 회귀(Regression)를 수행하는 모델을 뜻함

$$y = W_x + b$$

- 이때,  $x$ ,  $y$ 는 데이터를 의미하고,  $W$ 와  $b$ 는 데이터에 적합한 값으로 학습될 수 있는 파라미터(parameter) 임

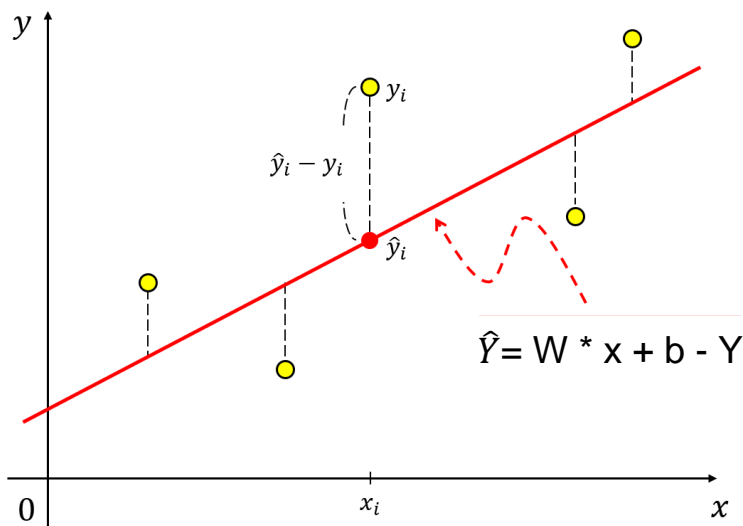


# 회귀의 성능평가 - MSE

- 제곱 오차(MSE; Mean Squared Error)

$$y = W_x + b$$

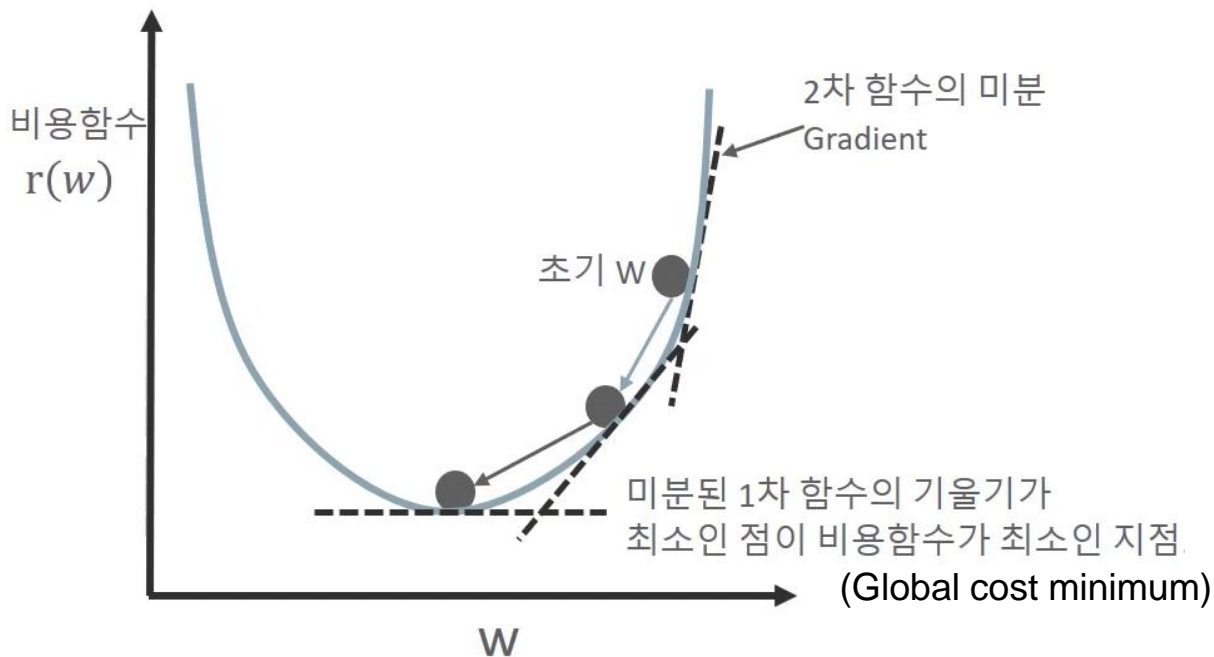
$$\sum (W \times x + b - Y)^2 / N$$



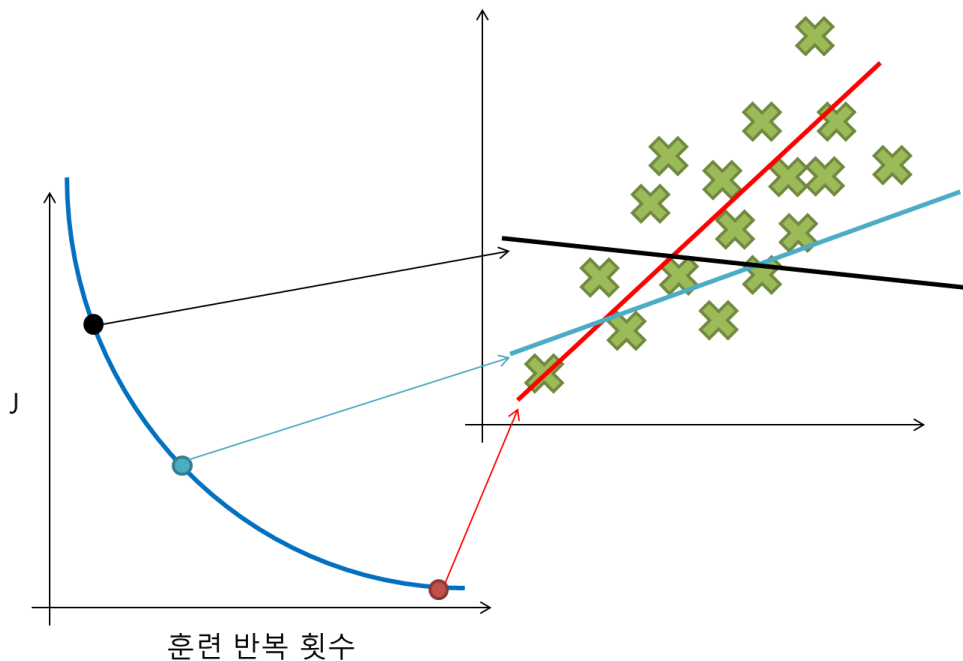
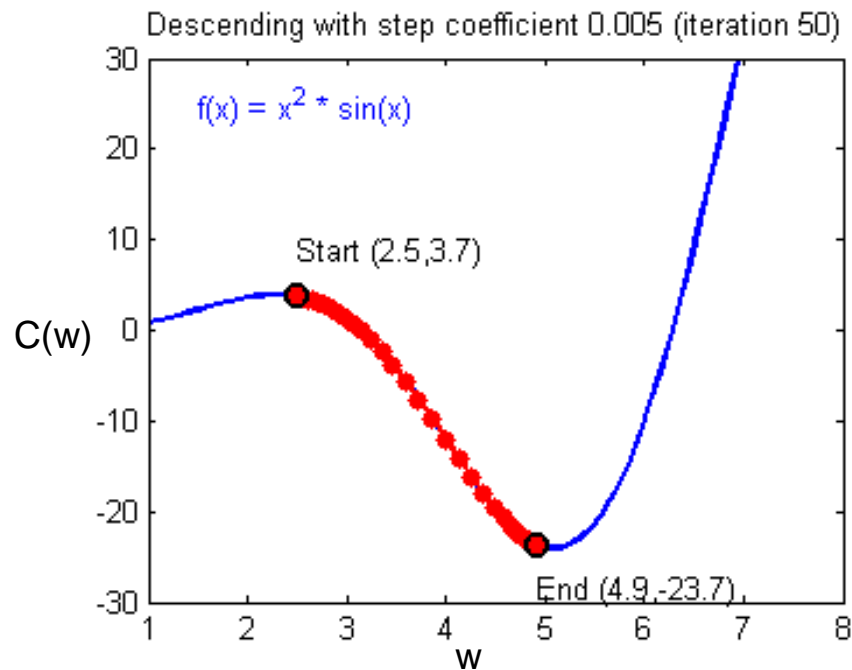
# 최적의 W를 찾는 것 최소 오차

- 머신러닝의 성능을 높이는 방법은? 오차를 최소화 하는 것

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



# 경사하강법(Gradient Descent)



<https://www.ibric.org/myboard/skin/news1/print.php?Board=news&id=280438>

## 회귀의 성능평가 지표 - MSE

- 손실(loss) = 비용(cost) = 에러(error)
- 손실함수 = 비용함수 = 에러함수
- loss function = cost function = error function



## 회귀의 성능평가 지표 - MSE

$$\text{손실함수(Loss Function)} = \sum (W \times x + b - Y)^2$$

$W \times x + b - Y$  (하나의  $x$ 에 대한 손실)

$\Rightarrow (W \times x + b - Y)^2$  의 전체 합

$\Rightarrow \sum (W \times x + b - Y)^2$  (모든  $x$ 의 손실 합)


데이터의 개수가 많아지면 손실이 커지므로  
전체 손실의 평균을 구함

$$\sum (W \times x + b - Y)^2 / N$$

## 회귀의 성능평가 - MSE

데이터의 개수가 많아지면 손실이 커지므로  
전체 손실의 평균을 구함

$$\sum (W \times x + b - Y)^2 / N$$

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


# 회귀의 성능평가 - MSE

## MSE 평가 지표 해석은 어떻게 할까?

- 정답  $y=[1, 10, 13, 7]$ 이고, 예측값  $\hat{y}=[10, 3, 1, 4]$ 와 같이 잘못 예측한다면 MSE 손실 함수는 아래와 같이 70.75라는 큰 값을 갖게 됨

$$MSE = \frac{1}{4}\{(10 - 1)^2 + (3 - 10)^2 + (1 - 13)^2 + (4 - 7)^2\} = 70.75$$

- 정답  $y=[1, 10, 13, 7]$ 이고, 예측값  $\hat{y}=[2, 10, 11, 6]$ 와 같이 비슷한 값을 예측한다면 MSE 손실 함수는 아래와 같이 1.5라는 값을 갖게 됨

$$MSE = \frac{1}{4}\{(2 - 1)^2 + (10 - 10)^2 + (11 - 13)^2 + (6 - 7)^2\} = 1.5$$

- 결과적으로 테스트 데이터에 대한 MSE가 작은 머신러닝 모델이 더욱 좋은 머신러닝 모델이라고 볼 수 있음.

# 회귀의 성능평가 - RMSE

## RMSE; Root Mean Squared Error

- MSE는 차이를 제곱해서 더하므로 차이가 증폭되는 문제가 있음.
- MSE에 Root를 씌운 형태, RMSE도 많이 사용됨

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

# 회귀의 성능평가 - MAE

## MAE; Mean Absolute Error

- 예측값과 정답 간의 차이에 절대값을 취함

$$MAE = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i|$$

# MSE, RMSE 함수

- MSE는 `mean_squared_error()` 함수 사용
- RMSE는 기본 함수로 제공하지 않기 때문에 **`np.sqrt`** 함수 이용

```
from sklearn.metrics import mean_squared_error
```

```
# MSE 평가
```

```
mse = mean_squared_error(y_test, y_pred)
```

```
# RMSE 평가
```

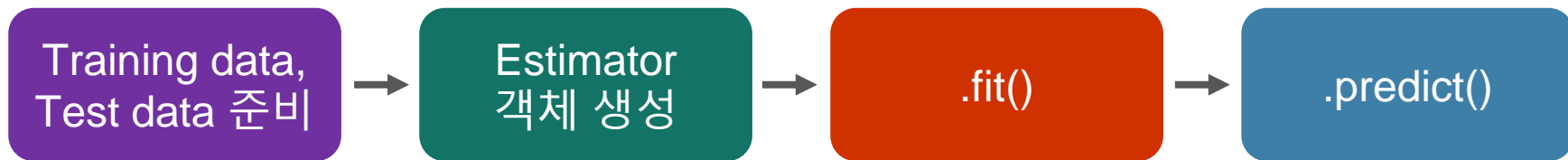
```
rmse = np.sqrt(mse)
```

---

# 회귀모델(Regression) 기본 익히기

# 회귀모델 머신러닝 수행 절차

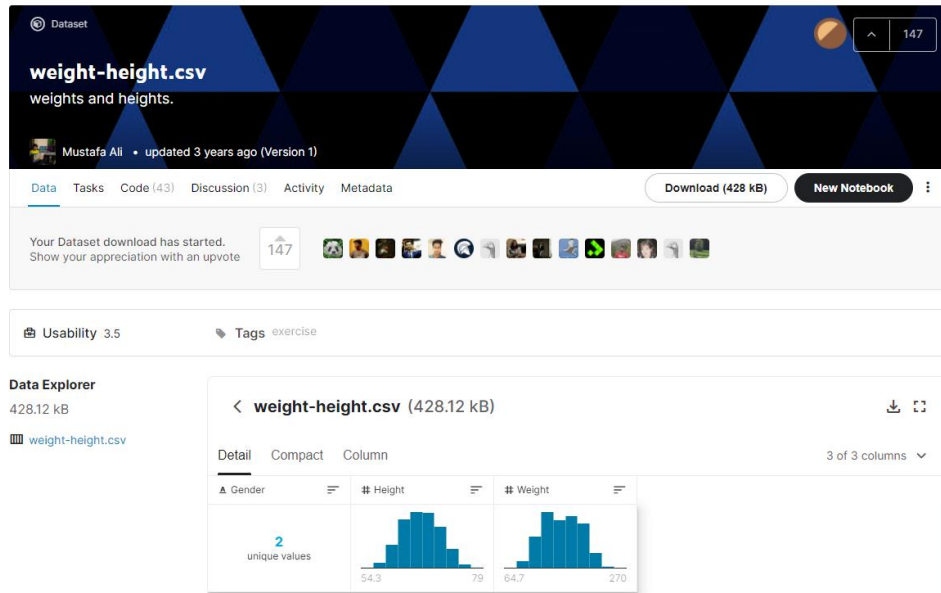
```
from sklearn.linear_model import LinearRegression  
model_lr = LinearRegression()  
model_lr.fit(X_train, y_train)  
prediction = model_lr.predict(x_test)
```





# [실습1] Linear Regression – 키로 몸무게 예측

- 키(Height)와 몸무게(Weight) 데이터
- <https://www.kaggle.com/mustafaali96/weight-height>
- 데이터 columns : Gender, Height, Weight
- 데이터 개수 : 10,000명

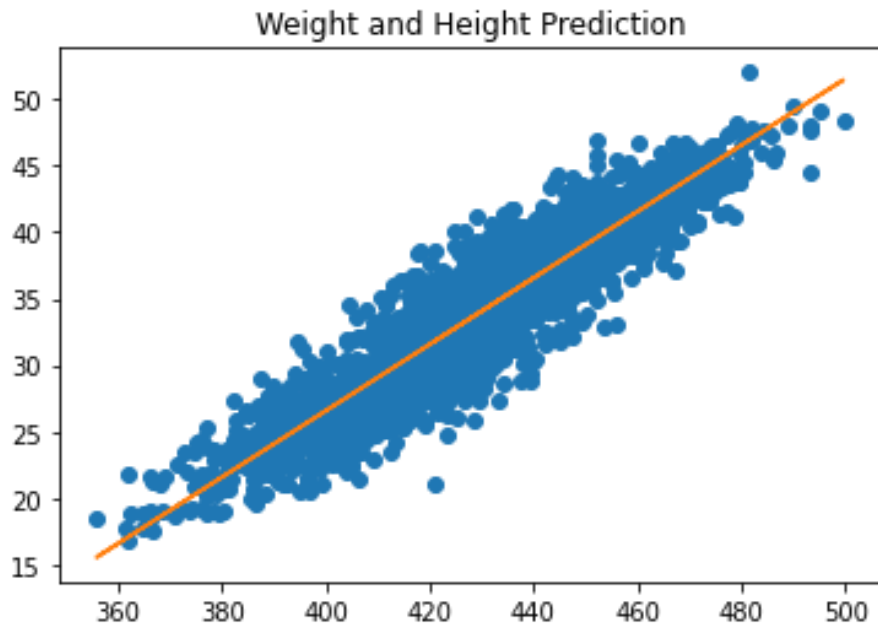


## [실습1] Linear Regression – 키로 몸무게 예측

- 예측모델 세팅-키(height) & 몸무게(Weight) 데이터
  - **feature** 데이터 : 사람의 키(Height)
  - **target** 데이터 : 사람의 몸무게(Weight)
  - 사용 알고리즘(Estimator) : LinearRegression

# [실습1] Linear Regression – 키로 몸무게 예측

- Linear Regression 키에 대한 몸무게 예측하기
- 3-3-1\_linear\_regression\_height\_weight\_pred.jpynb



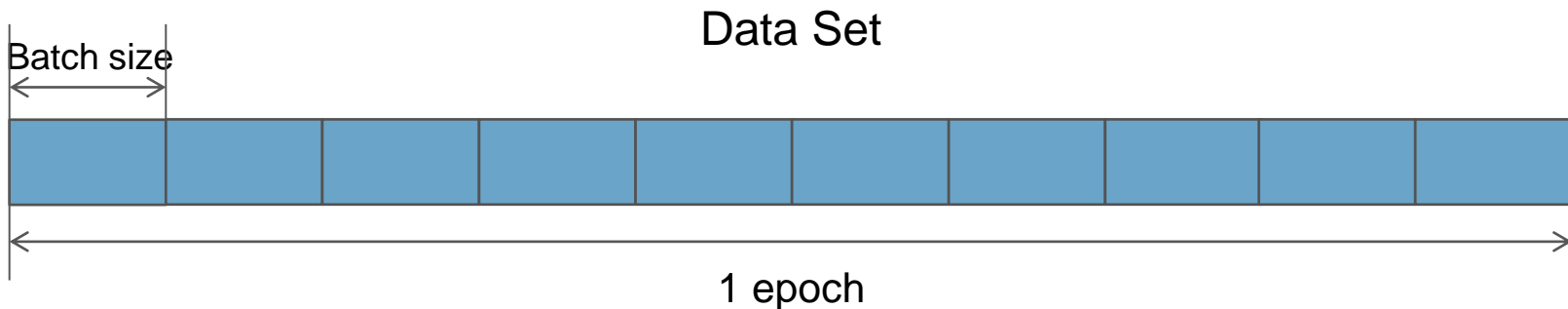
---

# Regression

[실습2]보스턴 부동산 가격 예측

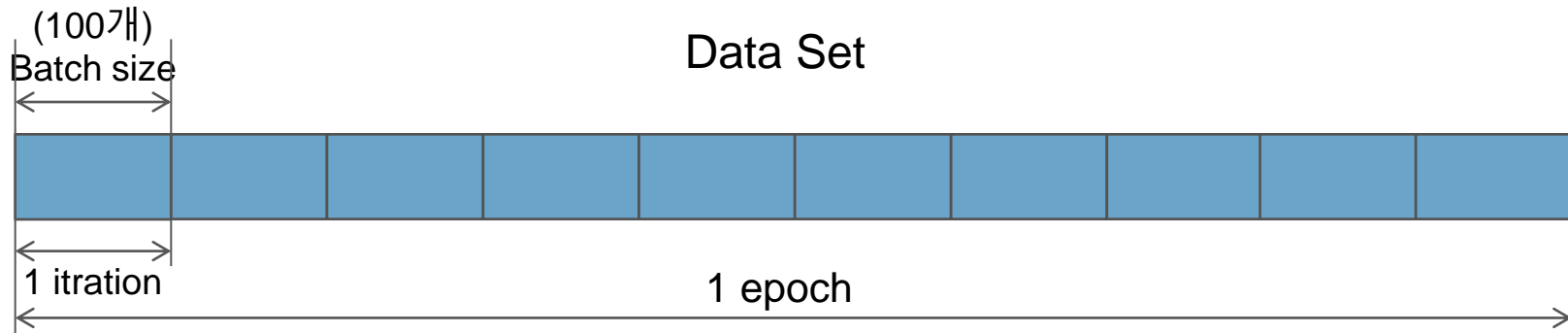
- k-Fold Cross Validation(교차검증)
- feature engineering

# 머신러닝 학습 이해



- Epoch(1회 학습)
  - 모델 학습에 포함되어 있는 모든 데이터 값들이 한번씩 모델에 들어 온 뒤 Weight(가중치) 값을 업데이트 하는 주기를 의미함
- Batch size
  - 여러 개의 batch(mini-batch)로 분할 할 때, 한 개의 batch안에 들어있는 데이터 갯 수
  - 한 batch안에서 cost가 가장 작은 weight를 찾아 업데이트 함
- Iteration(step)
  - 하나의 batch가 한번 학습이 이뤄짐을 의미함
  - 1 epoch를 진행하기 위해 Weight(가중치) 업데이트가 일어난 횟수

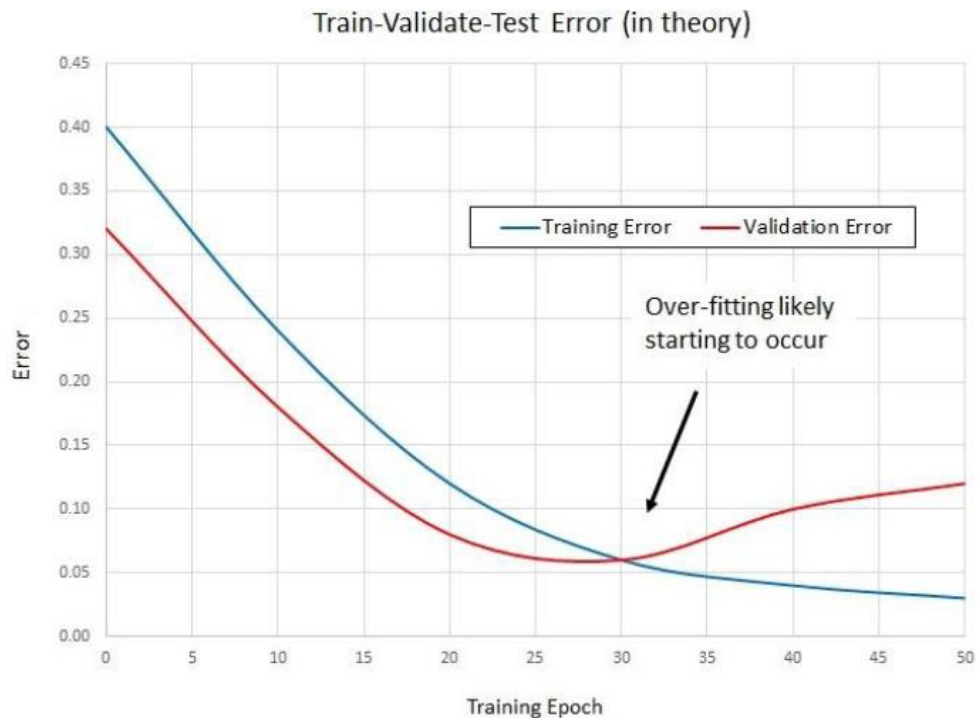
# (문제해결)머신러닝 학습 이해



- 총 데이터 개수 : 1000개
- batch size : 100개
- epochs : 20
- mini-batch 갯수? 10개
- 몇번의 iteration이 일어날까? 10번
- 총 iteration 횟수는? 200번

# train, validation

## Train data로 학습 / validation data로 모니터



# train, test, validation

Training set		Test set
학습을 위한 데이터 = Training Set (80%)	검증을 위한 데이터 = Validation Set(20%)	예측을 위한 데이터 = Test Set

- 과대적합, 과소적합 문제를 해결하기 위해 학습 시 검증을 위한 테스트 시행
- 특히 충분한 데이터가 없는 경우, 트레이닝 데이터가 어떻게 나뉘지는가에 따라 학습된 모델과 성능 측정결과가 크게 달라짐



# K-폴드 교차 검증(validation)

- 검증데이터 : 학습 데이터를 다시 분할하여 학습 데이터와 학습된 모델의 성능을 일차 평가하기 위한 데이터

학습 데이터 세트

분할

학습 데이터 세트

검증 데이터 세트



- 테스트 데이터 : 모든 학습/검증 과정이 완료된 후 최종적으로 성능을 평가하기 위한 데이터

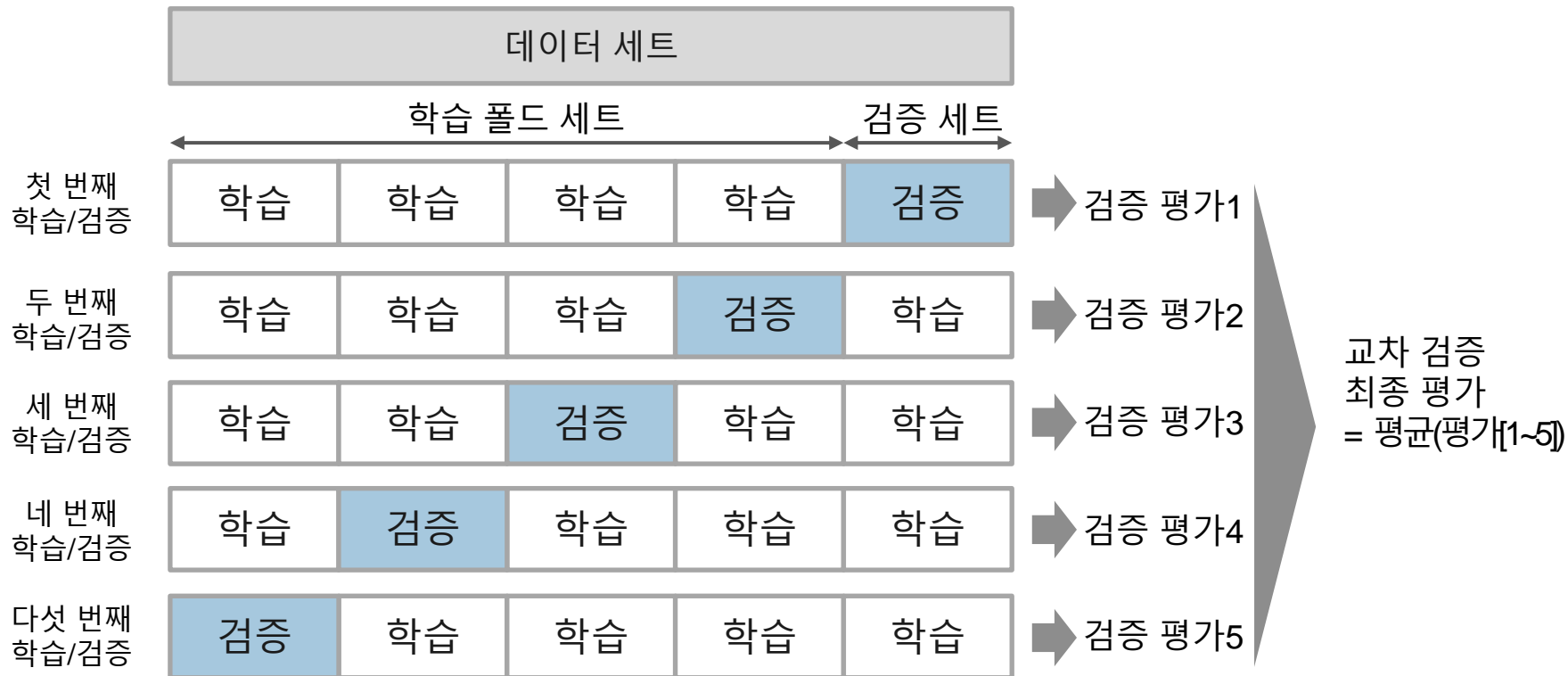
테스트 데이터 세트



수능

# K-폴드 교차 검증(validation)

- **k=5**; 총 5개의 폴드 세트에 5번의 학습과 검증 평가 반복 수행



# K-Fold Cross Validation 이해

- k-Fold Cross Validation 성능 보정
- k-Fold Cross Validation 구현 방법 참조

[https://scikit-](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html?highlight=kfold#sklearn.model_selection.KFold)

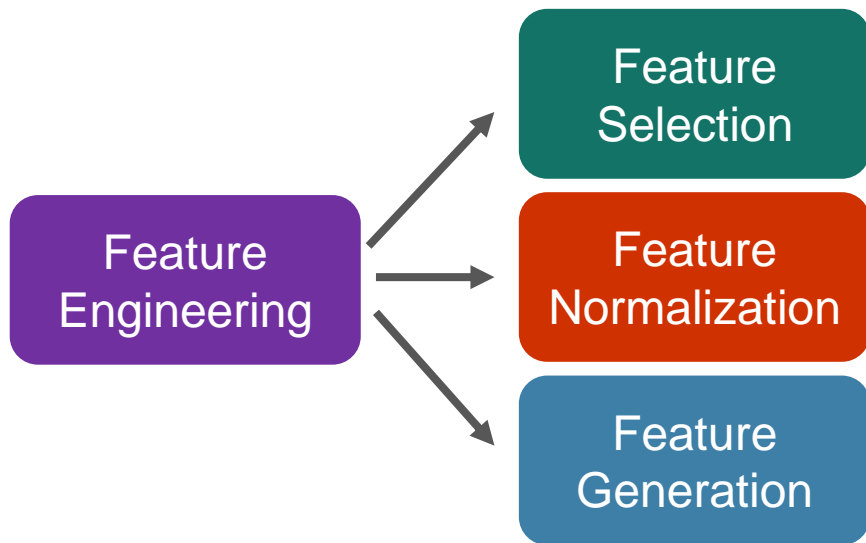
[learn.org/stable/modules/generated/sklearn.model\\_selection.KFold.html?highlight=kfold#sklearn.model\\_selection.KFold](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html?highlight=kfold#sklearn.model_selection.KFold)

```
>>> import numpy as np
>>> from sklearn.model_selection import KFold
>>> X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
>>> y = np.array([1, 2, 3, 4])
>>> kf = KFold(n_splits=2)
>>> kf.get_n_splits(X)
2
>>> print(kf)
KFold(n_splits=2, random_state=None, shuffle=False)
```

```
>>> for train_index, test_index in kf.split(X):
...     print("TRAIN:", train_index, "TEST:", test_index)
...     X_train, X_test = X[train_index], X[test_index]
...     y_train, y_test = y[train_index], y[test_index]
TRAIN: [2 3] TEST: [0 1]
TRAIN: [0 1] TEST: [2 3]
```

# Feature Engineering

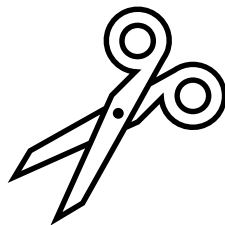
- 도메인 지식이나 분석을 통해서 유의미한 특징(Feature)들 만을 선별해내거나 Feature의 형태를 더욱 적합한 형태로 변경하는 것
- **적절한 Feature Engineering은 머신러닝 모델의 성능을 향상시킴**



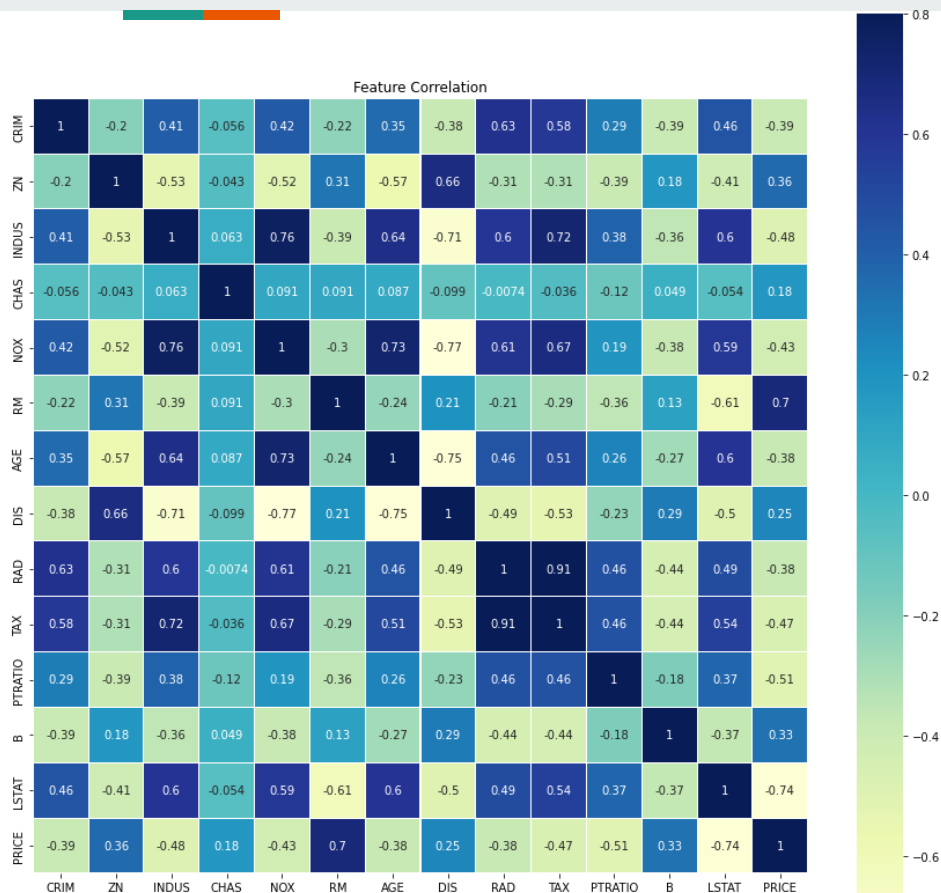
# Feature Engineering - Feature Selection

- 예측값과 연관이 없는 불필요한 특징을 제거해서 머신러닝 모델의 성능을 더욱 높이는 기법
- 제거할 특징을 선택하기 위해 상관분석(Correlation Analysis) 등을 진행함.

Feature  
Selection



# Feature Engineering - Feature Selection



## 상관분석(Correlation Analysis) 또는 상관관계 분석

- 두 Feature(독립변수) 간에 어떤 선형적 또는 비선형적 관계를 갖고 있는지를 파악하는 방법

1에 가까운 값 : 두 변수들 간의 양의 상관관계가 있음.

0에 가까운 값 : 두 변수들 간의 상관관계가 없음.

-1에 가까운 값 : 두 변수들 간의 음의 상관관계가 있음.

# Feature Engineering - Feature Selection

- 상관관계 분석 구현방법

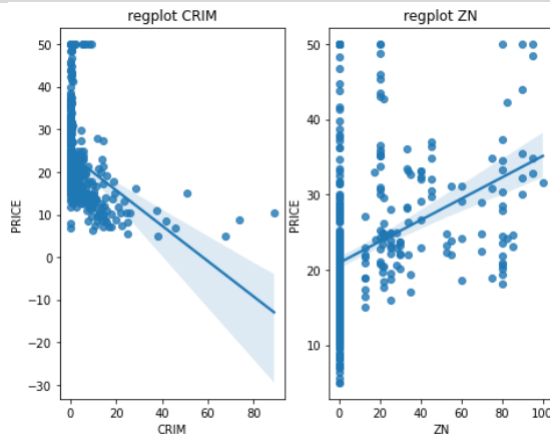
```
corr = df_boston_house.corr() # correlation 메소드 호출
plt.figure(figsize=(10, 10));
sns.heatmap(data=corr,
            vmax=0.8,
            linewidths=0.01,
            square=True,
            annot=True,
            cmap='YlGnBu');
plt.title('Correlation Matric')
```

# Feature Engineering - Feature Selection

## sns.regplot으로 Feature들 간의 경향성 출력하기

- `sns.regplot(data={dataframe}, x={컬럼명}, y={컬럼명})` 형태를 이용해서 **regression line**이 포함된 scatter plot를 그릴 수 있음.

```
sns.regplot(data = df_boston_house, x='RM', y='PRICE')  
plt.show()
```

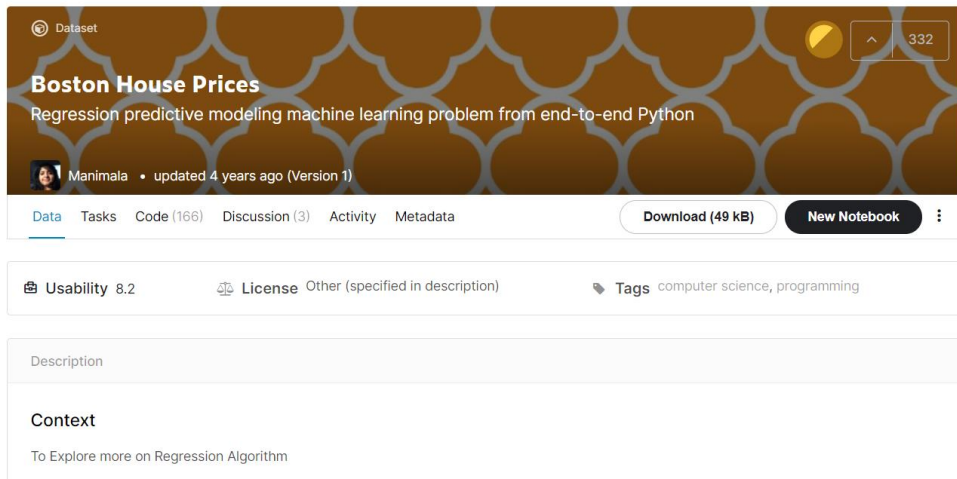




# [실습2] Regression – 집값 예측

## 보스턴 주택 가격(Boston House Prices) 회귀 모델 구현

- <https://www.kaggle.com/vikrishnan/boston-house-prices>
- 1970년대의 보스턴 지역의 부동산 가격을 수집한 데이터
- Feature 데이터 Columns : CRIM, RM, ... 13 Dimension
- 데이터 개수 : 506개



## [실습2] Regression – 집값 예측

- 보스턴 부동산 가격예측을 위한 데이터의 특징들(Features)
  - CRIM : 도시별 범죄발생률
  - ZN : 25,000평을 넘는 토지의 비율
  - INDUS : 도시별 비산업 지구의 비유
  - CHAS : 찰스 강의 더미 변수(1 = 강의 경계, 0 = 나머지)
  - NOX : 일산화질소 농도
  - RM : 주거할 수 있는 평균 방의개수
  - AGE : 1940년 이전에 지어진 주택의 비율
  - DIS : 5개의 고용지원센터까지의 가중치가 고려된 거리
  - RAD : 고속도로의 접근 용이성에 대한 지표
  - TAX : 10,000달러당 재산세 비율
  - PTRATIO : 도시별 교사와 학생의 비율
  - B : 도시의 흑인 거주 비유
  - LSTAT : 저소득층의 비율

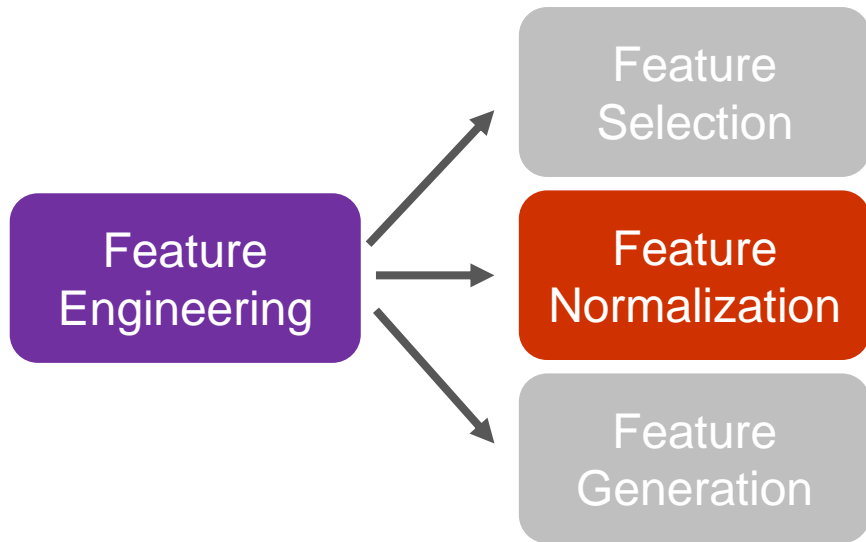
## [실습2] Regression – 집값 예측

Regression으로 boston house price 예측하기

- **input data** : CRIM, RM, .... 13개
- **target data** : 보스턴 부동산 집값(단위: \$1000)
- 사용 알고리즘(Estimator) : LinearRegression
- 추가활용 기능 : Feature Selection 적용 성능 비교
- 실습 파일
  - 3-3-2.Regression\_boston\_house\_price\_pred(EDA\_Feature Selection).ipynb

# Feature Engineering

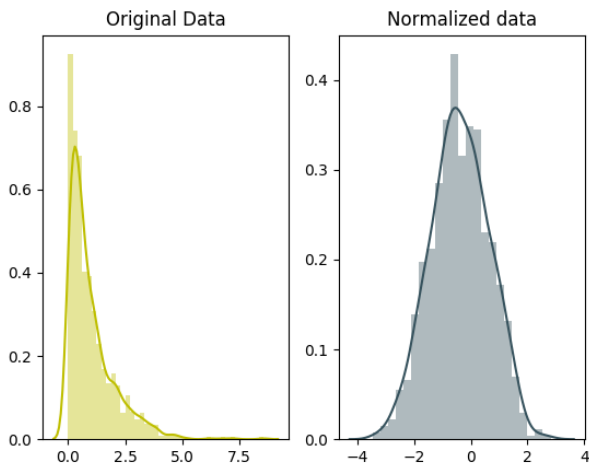
- 도메인 지식이나 분석을 통해서 유의미한 특징(Feature)들 만들 선별해내거나 **Feature의 형태를 더욱 적합한 형태로 변경하는 것**
- 적절한 Feature Engineering은 머신러닝 모델의 성능을 향상시킴



# Feature Engineering – Feature Normalization

- **Standardization(표준정규분포)**

- 값의 분포를 정규분포(Normal Distribution) 형태로 변경
- **평균 0, 표준편차를 1로 맞춤**
- 일반적으로 **Feature 값에 대한 정규화를 수행할 경우, 더 안정적인 머신러닝 모델을 학습시킬 수 있음.**  $\mu$  : 평균,  $\sigma$  : 표준편차



$$x_{new} = \frac{x - \mu}{\sigma}$$
$$= \frac{x - \text{mean}(x)}{\text{stdev}(x)}$$

# Feature Engineering – Feature Normalization

- **Min-Max Scaling**

- 값의 분포를 정규분포(Normal Distribution) 형태로 변경
- 0~1 사이의 분포로 조정

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

# Feature Engineering – Feature Normalization



- **Standardization 활용 방법**

- StandardScaler 클래스를 이용해서 데이터 정규화(Normalize) 하기

```
from sklearn.preprocessing import StandardScaler
```

```
normalized_data = StandardScaler().fit_transform(data)
```

# Feature Engineering – Feature Normalization

- **Min-Max Scaling 활용 방법**

- MinMaxScaler 클래스를 이용해서 데이터 정규화(Normalize) 하기

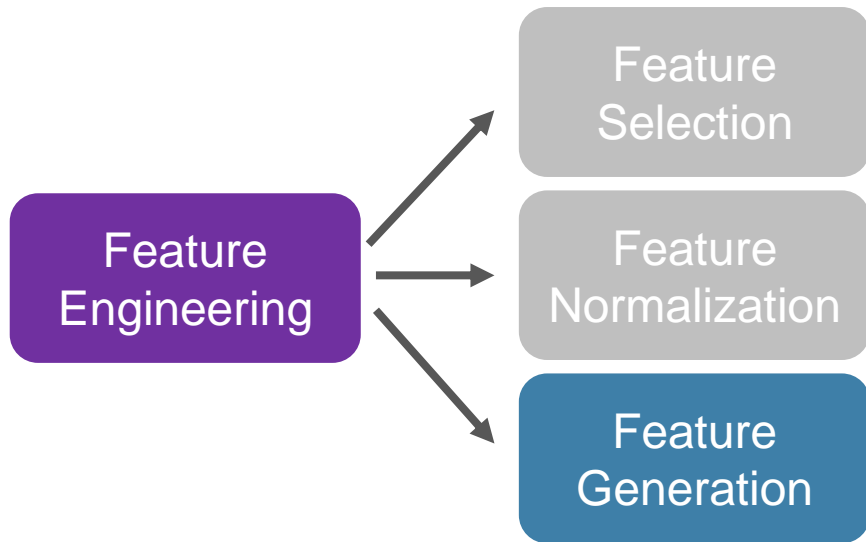
```
from sklearn.preprocessing import MinMaxScaler
```

```
normalized_data = MinMaxScaler().fit_transform(data)
```



# Feature Engineering

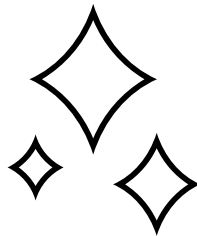
- 도메인 지식이나 분석을 통해서 유의미한 특징(Feature)들 만들 선별해내거나 **Feature의 형태를 더욱 적합한 형태로 변경하는 것**
- 적절한 Feature Engineering은 머신러닝 모델의 성능을 향상시킴



# Feature Engineering – Feature Generation

- Feature 값들을 조합해서 새로운 특징을 만들어 내는 Feature Engineering 기법

Feature  
Generation



## Polynomial Features

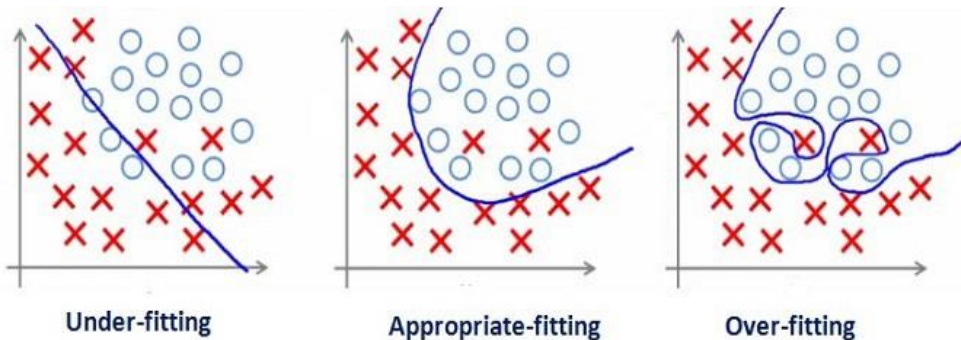
- 서로 다른 Feature들 간의 곱셈으로 새로운 Feature로 만들 수 있음.
- 즉, CRIM(범죄율)  $x_1$  과 LSTAT(저소득층 비율)  $x_2$  2개의 특징으로 CRIM x LSTA(  $x_1 \times x_2$  )라는 새로운 특징을 만들어 낼 수 있음.

---

# Regression의 Overfitting 문제해결

# Overfitting, Underfitting

- 오버피팅(Overfitting)
  - 학습 과정에서 머신러닝 알고리즘의 파라미터가 트레이닝 데이터에 과도하게 최적화되어 트레이닝 데이터에 대해서는 잘 동작하지만 새로운 데이터인 테스트 데이터에 대해서는 잘 동작하지 못하는 현상
  - 오버피팅을 방지하기 위한 기법을 Regularization 이라고 함.
- 언더피팅(Underfitting)
  - 모델의 표현력이 부족해서 트레이닝 데이터도 제대로 예측하지 못하는 상황



오버 피팅을 방지하기  
위한 기법:  
Regularization 기법

# 기본 Linear Regression 확장 알고리즘

- 기본 Linear Regression의 Overfitting 문제 발생 해결을 위해
- 기본 Linear Regression을 확장한 다음과 같은 알고리즘들이 있음.
- 기본 Linear Regression : **Regularization**을 적용하지 않은 알고리즘
- **Ridge** : L2 Regularization을 적용한 모델
- **Lasso** : L1 Regularization을 적용한 모델
- **ElasticNet** : L1 Regularization과 L2 Regularization을 함께 적용한 모델

# Linear Regression Estimator 선언하기

- 선형회귀(Linear Regression) Estimator 선언하기

```
from sklearn.linear_model import LinearRegression
```

```
lr = LinearRegression()
```

MSE가 최소가 되게 하는 가중치와 편향을 찾음.

# Ridge Regression Estimator 선언하기

- 기본 Linear Regression에서 발생하는 Overfitting 문제를 **L2 Regularization**을 이용해서 **가중치  $w$ 가 너무 커지지 않도록 장려하는 Regression 기법.**
- 선형회귀(Ridge Regression) Estimator 선언하기

```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=1.0)
# alpha값이 크면 regularization 효과가 커지는 쪽으로 적용,
# alpha값이 작으면 효과가 작아지는 쪽으로 적용됨
```

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.Ridge.html?highlight=ridge#sklearn.linear\\_model.Ridge](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html?highlight=ridge#sklearn.linear_model.Ridge)



# Lasso Regression Estimator

- 기본 Linear Regression에서 발생하는 Overfitting 문제를 개선하기 위해서 **L1 Regularization**을 이용해서 **가중치  $w$ 가 너무 커지지 않도록 장려하는 Regression** 기법임.
- 선형회귀(Lasso Regression) Estimator 선언하기

```
from sklearn.linear_model import Lasso
Lasso_reg = Lasso(alpha=1.0)
# alpha값이 크면 regularization 효과가 커지는 쪽으로 적용,
# alpha값이 작으면 효과가 작아지는 쪽으로 적용됨
```

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.Lasso.html?highlight=lasso#sklearn.linear\\_model.Lasso](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html?highlight=lasso#sklearn.linear_model.Lasso)

# ElasticNet Regression Estimator 선언하기

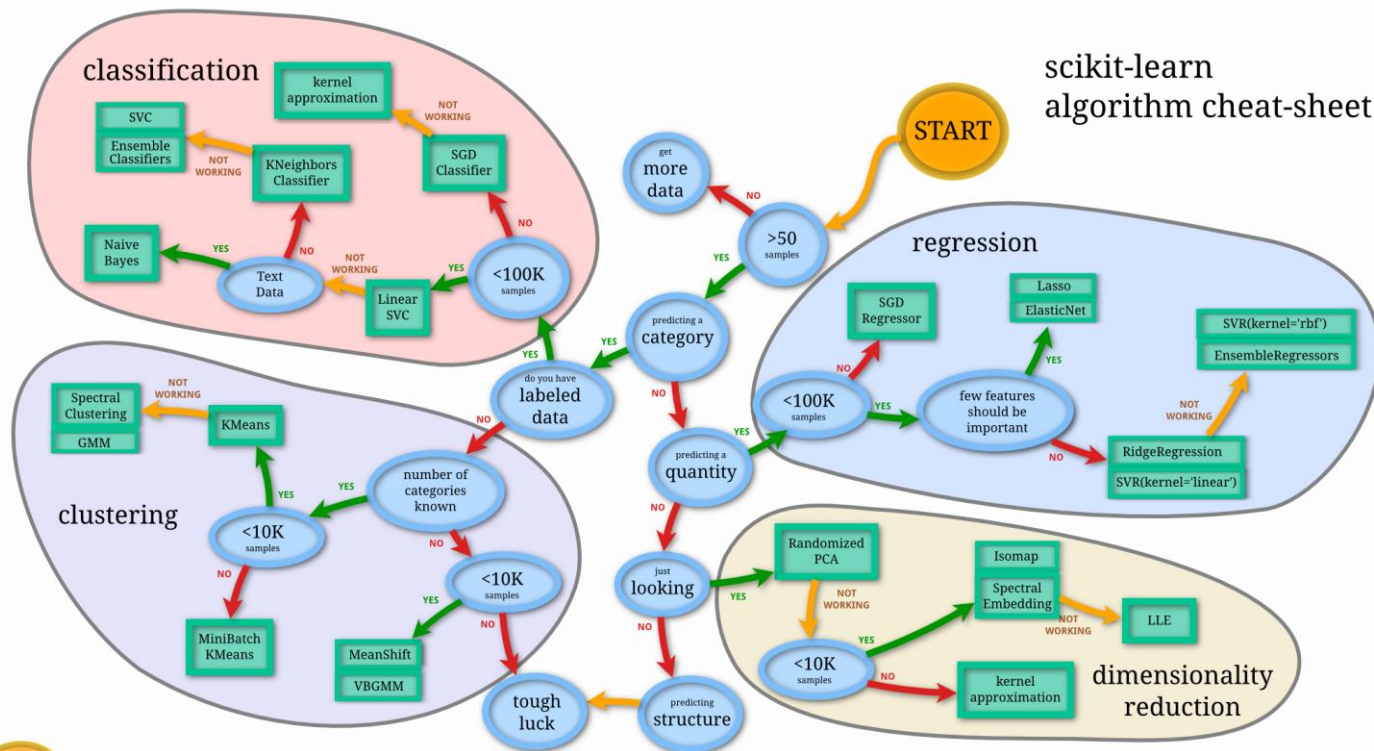
- 기본 Linear Regression에서 발생하는 Overfitting 문제를 개선하기 위해서 **L1+L2 Regularization**을 이용해서 **가중치  $w$ 가 너무 커지지 않도록** 장려하는 Regression 기법임.
- 선형회귀(ElasticNet Regression) Estimator 선언하기

```
from sklearn.linear_model import ElasticNet
elasticnet_reg = ElasticNet(alpha=1.0)
# alpha값이 크면 regularization 효과가 커지는 쪽으로 적용,
# alpha값이 작으면 효과가 작아지는 쪽으로 적용됨
```

[https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.ElasticNet.html?highlight=elasticnet#sklearn.linear\\_model.ElasticNet](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html?highlight=elasticnet#sklearn.linear_model.ElasticNet)

# Scikit-learn을 이용해서 선형회귀 Estimator 선언하기

[https://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/index.html](https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html)



# 하이퍼 파라미터(Hyper-Parameter) 튜닝

- 학습과정에서 알고리즘에 의해 변경되는 파라미터 외에 알고리즘 디자이너가 설정해 줘야만 하는 값
- 적절한 **하이퍼 파라미터 값**을 찾아서 설정해 주는 것도 **머신러닝의 중요요소 중 하나**임.

- scikit-learn 공식문서 참고

[https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear\\_model](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model)

## [실습3] Regression – 집값 예측 – 성능 비교

- Regression으로 boston house price 예측 성능 향상 시키기

실습 : [완성]3-3-3.Reggression\_boston\_house\_price\_(Advanced\_Estimator)\_ipynb

- 사용 알고리즘(Estimatior)

- LinearRegression
- Ridge
- Lasso
- ElasticNet

- 추가적인 적용 기법

- Feature Normalization
- Hyper-Parameter 튜닝