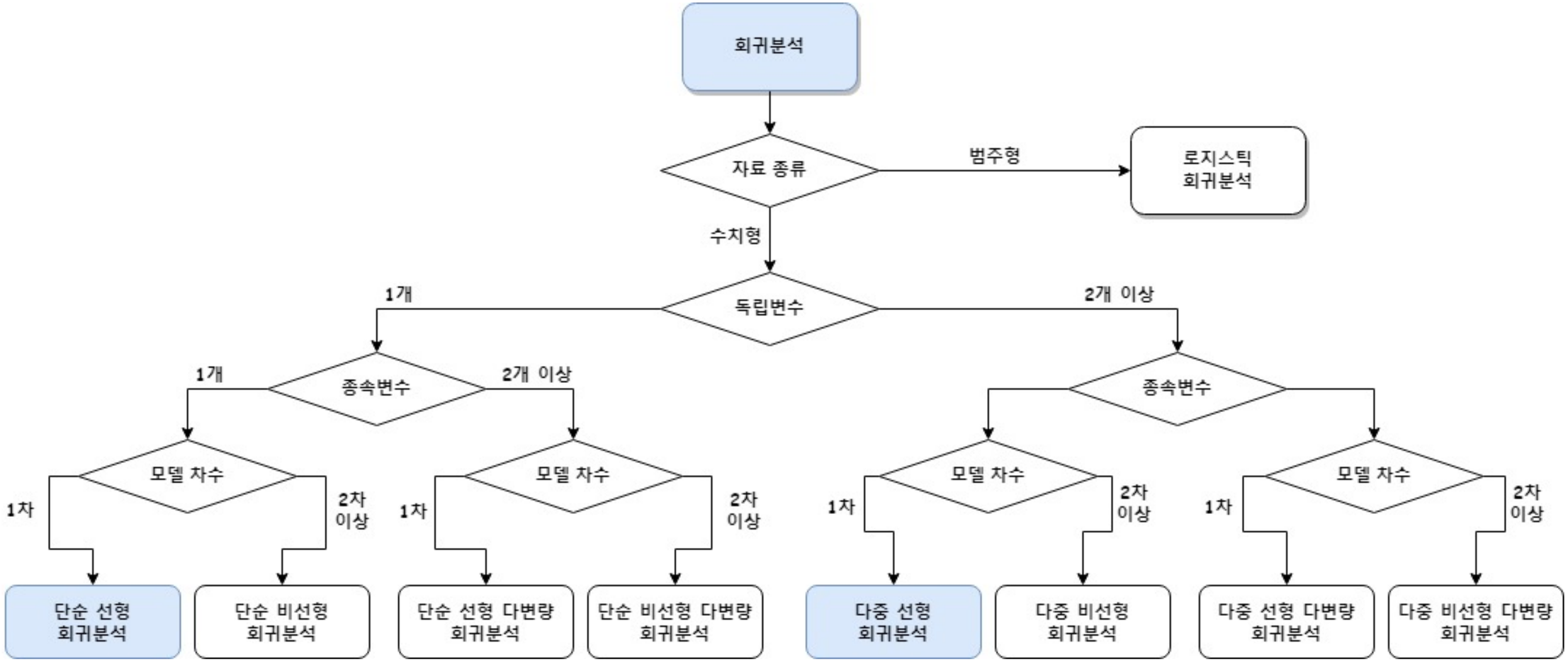
# AI Programming

# AI 프로그래밍-다중분류/회귀

2024.06.28
국립안동대학교
컴퓨터교육과 PhD. 조영복
(ybcho@anu.ac.kr)

# 단순 / 다중 선형 회귀분석을 가장 많이 사용



# 분류(classification)

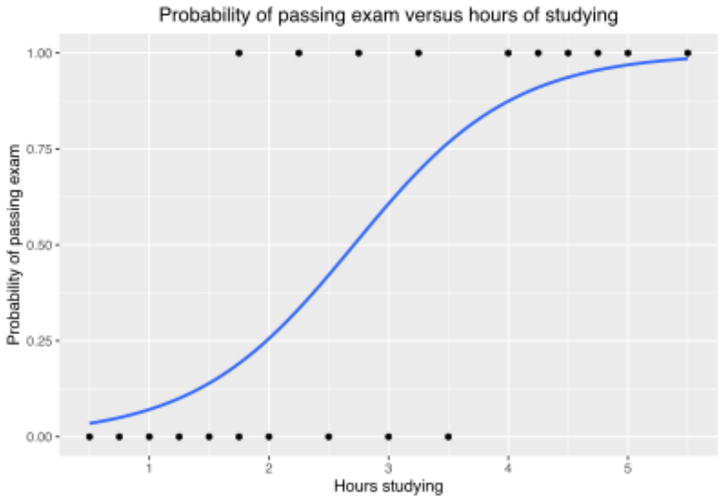
- 데이터셋으로부터 특징들을 파악하여 모델을 학습시키고, 분류 결과를 도출
- 분류로 해결할 수 있는 대표적인 문제들을 예로 들자면,
  - 타이타닉 데이터셋으로부터 생존자 예측하기 (생존 or 사망)
  - 고객들의 금융 데이터셋으로부터 대출 가능 여부 예측하기 (가능 or 불가능)
  - 1~10까지의 숫자 이미지가 들어있는 MNIST 데이터셋으로부터 해당 숫자가 몇인지 분류하기

# 분류(classification)

- 분류 알고리즘
  - Logistic Regression (로지스틱 회귀모형)
  - Stochastic Gradient Descent (확률적 경사 하강법)
  - K-nearest neighbor (k-최근접 이웃)
  - Decision Tree (의사결정나무)
  - Support Vector Machine (서포트 벡터 머신)

# 분류(classification)

- 분류 알고리즘
  - Logistic Regression (로지스틱 회귀모형)
    - 2진분류기에 사용가능
    - 어떤 사건이 발생될 확률로 0에서 1 사이의 값
    - 경계값 0.5를 기준으로 0.5보다 크다면 양성 클래스, 0.5보다 작다면 음성 클래스로 분류



# 회귀분석(regression analysis)

- 회귀
  - 부모와 자식 간의 키의 상관관계를 분석
    - 부모의 키가 모두 클 때 자식의 키가 크긴 하지만 그렇다고 부모를 능가할 정도로 크지 않았고,
    - 부모의 키가 모두 아주 작을 때 그 자식의 키가 작기는 하지만 부모보다는 큰 경향
    - 부모의 키가 아주 크더라도 자식의 키가 부모보다 더 커서 세대를 이어가면서 무한정 커지는(발산) 것은 아니며, 부모의 키가 아주 작더라도 자식의 키가 부모보다 더 작아서 세대를 이어가며 무한정 작아지는(수렴) 것이 아니라는 것이다. 즉, 사람의 키는 평균 키로 회귀하려는 경향을 가진다는 자연의 법칙.
  - 회귀는
    - **여러 개의 독립변수**와 **한 개의 종속변수** 간의 상관관계를 모델링하는 기법
    - 아파트가격?
      - 망 개수, 방크기, 주변 학군등에 영향을 받는가?

# 회귀분석(regression analysis)

- 머신러닝 회귀예측
  - 독립변수는 피처에 해당되며 종속변수는 결정 값
  - 주어진 피처와 결정 값 데이터 기반에서 학습을 통해 **최적의 회귀 계수**를 찾아내는 것
  - 회귀유형
    - 회귀계수의 선형/비선형
    - 독립변수의 개수
    - 종속변수의 개수

- 회귀에서 가장 중요한 것
  - 회귀 계수
  - 이 회귀 계수가 선형인지 아닌지에 따라 선형 회귀와 비선형 회귀

# 회귀분석(regression analysis)

- 지도학습 : 분류와 회귀
  - 분류 : 예측값이 카테고리와 같은 이산형 클래스 값
  - 회귀 : 연속형 숫자 값

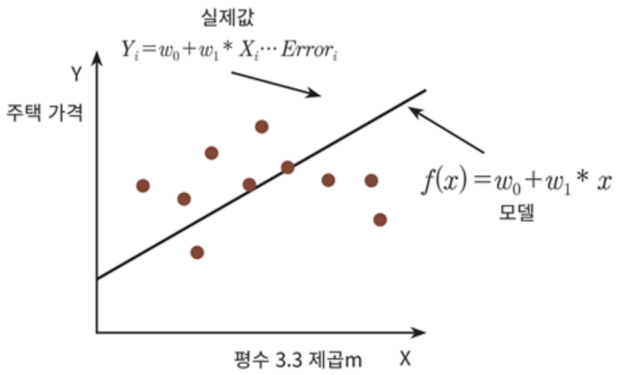
- 선형 회귀는
  - 실제 값과 예측값의 차이(오류의 제곱 값)를 최소화하는 직선형 회귀선을 최적화하는 방식
  - 선형 회귀 모델은 규제(Regularization) 방법에 따라 다시 별도의 유형으로 구분
  -

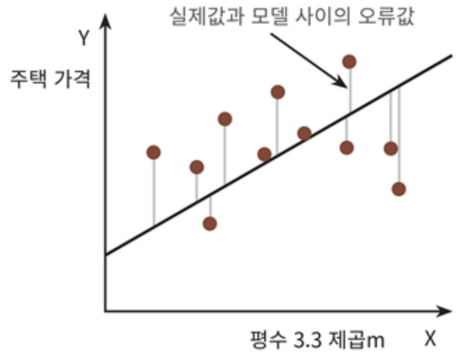
# 선형회귀 모델

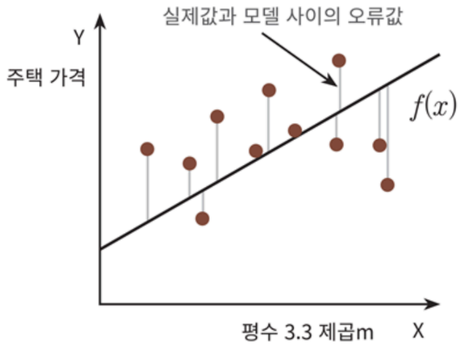
- 일반 선형 회귀
- 릿지(Ridge)
- 라쏘(Lasso) :
- 엘라스틱넷(LeasticNet)
- 로지스틱 회귀(Logistic Regression)

# 선형회귀 모델

- 선형 회귀
  - 독립변수가 1개인 단순 선형 회귀에서는 기울기 w1과 절편 w0를 회귀계수




RSS = (#1 주택 가격 – $(w_0+w_1$ #1 주택크기)$^2$
+ (#2 주택 가격 – $(w_0+w_1$ #2 주택크기)$^2$
+ (#3 주택 가격 – $(w_0+w_1$ #3 주택크기)$^2$
+ … (모든 학습 데이터에 대해 RSS 수행)

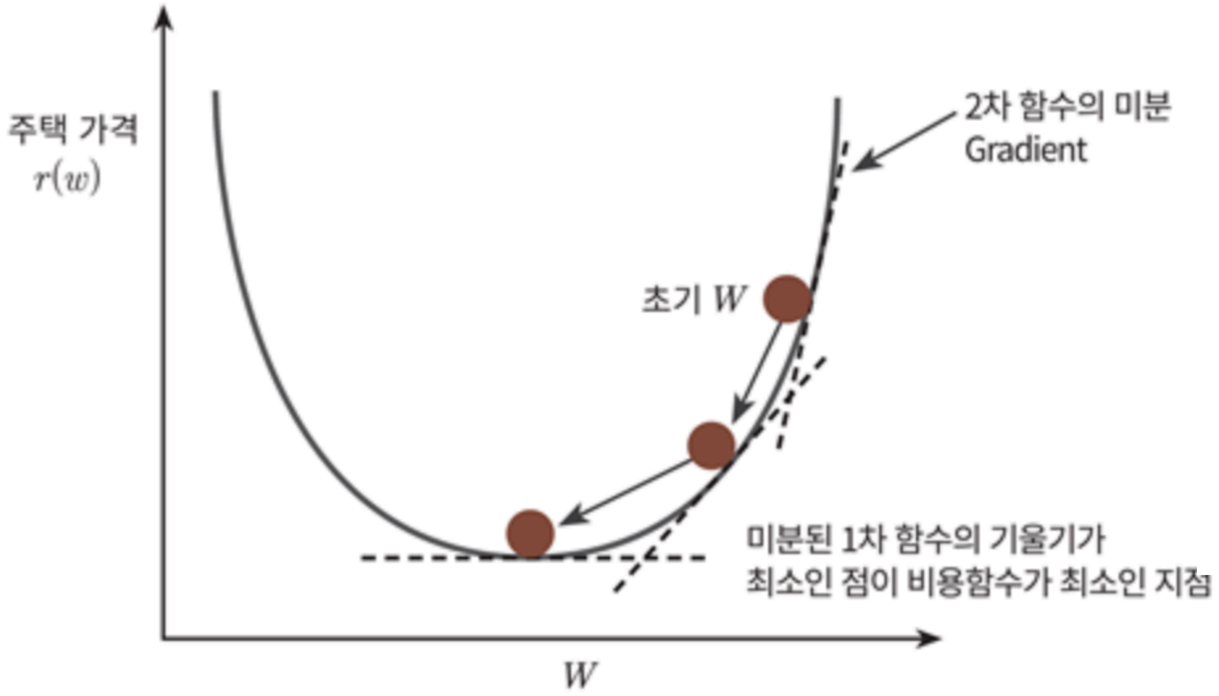
# 선형회귀 모델

- 일반 선형 회귀
  - 예측값과 실제값의 RSS(Residual Sum of Squares)를 최소화할 수 있도록 회귀 계수를 최적화하며, 규제(Regularization)를 적용하지 않은 모델.
- 릿지(Ridge)
  - 선형 회귀에 L2 규제를 추가한 회귀 모델
  - L2 규제를 적용, L2 규제는 회귀 계수 값의 예측 영향도를 감소시키기 위해서 회귀 계수값을 더 작게 만드는 규제 모델.
- 라쏘(Lasso)
  - 선형 회귀에 L1 규제를 적용한 방식.
  - L2 규제가 회귀 계수 값의 크기를 줄이는 데 반해,  L1 규제는 예측 영향력이 적은 피처의 회귀 계수를 0으로 만들어 회귀 예측 시 피처가 선택되지 않게 하는 것.
  - 이러한 특성 때문에 **L1 규제는 피처 선택 기능.**
- 엘라스틱넷(LeasticNet)
  - L2, L1 규제를 함께 결합한 모델
  - 주로 피처가 많은 데이터 셋에서 적용되며, L1 규제로 피처의 개수를 줄임과 동시에 L2 규제로 계수 값의 크기를 조정.
- 로지스틱 회귀(Logistic Regression)
  - 분류에 사용되는 선형 모델로 매우 강력한 분류 알고리즘
  - 이진 분류뿐만 아니라 희소 영역의 분류(예를 들어 텍스트 분류와 같은 영역에서 뛰어난 예측 성능).

# 선형회귀 모델

- **잔차(오류값)** 합을 구하는 방법은 2가지를 사용,
  - MAE (Mean Absolute Error) : 절대값을 취하는 방법
  - RSS  (Residual Sum of Square) : 제곱을 취하는 방법

# 경사하강법




**weight의 업데이트** = **에러 낮추는 방향 (decent)** X **한발자국 크기 (learning rate)** X **현 지점의 기울기 (gradient)**

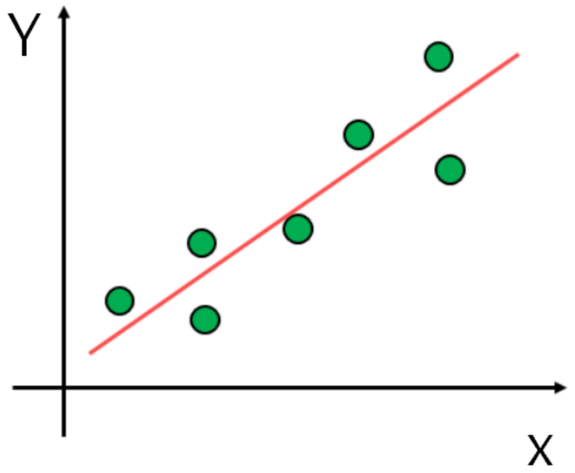
$$-\gamma\nabla F(\mathbf{a^n}) \quad = \quad - \quad \gamma \quad \nabla F(\mathbf{a^n})$$

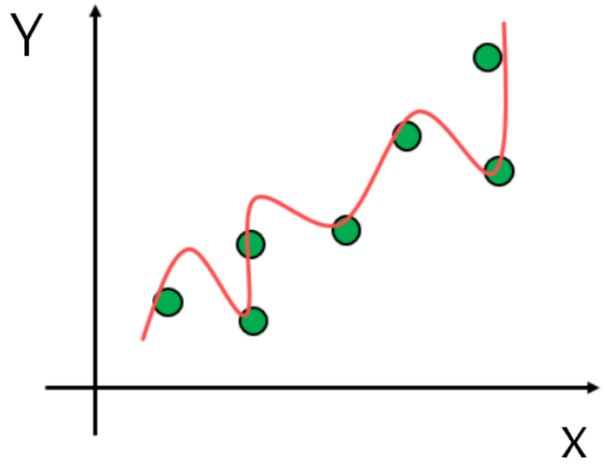
$$\frac{\partial R(w)}{\partial w_1} = \frac{2}{N}\sum_{i=1}^{N} -x_t * (y_i - (w_0 + w_1 x_i) = -\frac{2}{N}\sum_{i=1}^{N} x_i * (\text{실제값}_i - \text{예측값}_i)$$

$$\frac{\partial R(w)}{\partial w_0} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (w_0 + w_1 x_i) = -\frac{2}{N}\sum_{i=1}^{N} (\text{실제값}_i - \text{예측값}_i)$$

# Regularization = 정규화 = 제약 (penalty)

- 잔차의 경우는 train data에 대해서만 고려한 오류
- train data에 해당하는 잔차를 줄이기 위해서는 복잡한 회귀식이 도출
- 복잡한 회귀식으로 train data에 과적합되면 될수록 실제데이터에서는 예측력이 떨어지는 현상이 발생




Overfitting 은
학습데이터에 너무 최적화되어 W 값이 잡히고,
이후 학습 데이터가 아닌 새로운 데이터에는
올바른 값을 내보내지 못하는 현상을 의미

# L1 Regularization

- W의 절대값에 패널티를 부여하는 방식
- 영향력이 크지 않는 회귀계수값을 0으로 변환 = feature selection 기능
- Lasso 회귀
  - W의 절대값에 패널티를 부여하는 방식
  - 영향력이 크지 않는 회귀계수값을 0으로 변환 = feature selection 기능
  - Lasso 회귀

$$\text{비용함수 목표} = \text{Min}(RSS(W) + alpha * ||w||_1)$$

1) training accuracy  2) Generalization accuracy

- L2 Regularization
  - W의 제곱에 패널티를 부여하는 방식
  - 회귀계수의 크기를 감소시킴
  - Ridge 회귀
  -

$$\text{비용함수 목표} = \text{Min}(RSS(W) + alpha * ||w||_2^2)$$

1) training accuracy  2) Generalization accuracy

# L1 Regularization

```
# alpha값에 따른 회귀 모델의 폴드 평균 RMSE를 출력하고 회귀 계수값들을 DataFrame으로 반환
def get_linear_reg_eval(model_name, params=None, X_data_n=None, y_target_n=None,
                        verbose=True, return_coeff=True):
    coeff_df = pd.DataFrame()
    if verbose : print('####### ', model_name , '#######')
    for param in params:
        if model_name =='Ridge': model = Ridge(alpha=param)
        elif model_name =='Lasso': model = Lasso(alpha=param)
        elif model_name =='ElasticNet': model = ElasticNet(alpha=param, l1_ratio=0.7)
        neg_mse_scores = cross_val_score(model, X_data_n,
                                         y_target_n, scoring="neg_mean_squared_error", cv = 5)
        avg_rmse = np.mean(np.sqrt(-1 * neg_mse_scores))
        print('alpha {0}일 때 5 폴드 세트의 평균 RMSE: {1:.3f} '.format(param, avg_rmse))
        # cross_val_score는 evaluation metric만 반환하므로 모델을 다시 학습하여 회귀 계수 추출

        model.fit(X_data_n , y_target_n)
        if return_coeff:
            # alpha에 따른 피처별 회귀 계수를 Series로 변환하고 이를 DataFrame의 컬럼으로 추가.
            coeff = pd.Series(data=model.coef_ , index=X_data_n.columns )
            colname='alpha:'+str(param)
            coeff_df[colname] = coeff

    return coeff_df
# end of get_linear_regre_eval

# 라쏘에 사용될 alpha 파라미터의 값들을 정의하고 get_linear_reg_eval() 함수 호출
lasso_alphas = [ 0.07, 0.1, 0.5, 1, 3]
coeff_lasso_df =get_linear_reg_eval('Lasso', params=lasso_alphas, X_data_n=X_data, y_target_n=y_target)
```

# 선형회귀 모델

- 선형 회귀 모델을 적용하기 전에 먼저 데이터에 대한 스케일링/정규화 작업을 수행하는 것
  - 피쳐데이터(x)
  - case1) StandardScaler 클래스를 이용해 평균 0, 분산 1인 표준 정규분포를 가진 데이터 세트로 변환
  - case2) MinMaxScaler 클래스를 이용해 최솟값이 0이고, 최댓값이 1인 값으로 정규화를 수행
  - case3) log 변환
  - case4) categorical variable은 label encoding이 아닌 one-hot encoding 수행

- 타킷값(y)
  - 일반적으로 log변환
    - -> log(0)는 무한대값이 되므로, log(관측값+1)에 해당하는 numpy.log1p()를 사용
    - -> 예측값은 log1p()를 적용하여 예측한 것으므로, 최종예측값은 numpy.expm1() 을 적용해야 한다.
    -

# 머신러닝 회귀분석

- 연속적인 값 예측. 가격, 매출, 주가 등등의 연속성이 있는 데이터의 예측에 사용되는 알고리즘
  - 설명(독립)변수 (예측에 사용되는 변수) → 독립 변수 학습(머신러닝알고리즘 : 회귀분석) →예측(종속)변수
  - 단순회귀분석 : 독립변수 X, 종속변수 Y (독립변수와 종속변수가 1:1) →Y = aX + b
    - auto-mpg.csv (https://archive.ics.uci.edu/dataset/9/auto+mpg)

# 머신러닝 회귀분석

```
# 기본 라이브러리 불러오기
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 데이터 가져오기
df = pd.read_csv("auto-mpg.csv", header=None)

# 열 이름 지정
df.columns=['mpg','cylinders','displacement',
            'horsepower','weight','acceleration',
            'model year','origin','name']
df.head()
```

| | mpg | cylinders | displacement | ... | model year | origin | n |
|---|---|---|---|---|---|---|---|
| 0 | 18.0 | 8 | 307.0 | ... | 70 | 1 | chevrolet chevelle mal |
| 1 | 15.0 | 8 | 350.0 | ... | 70 | 1 | buick skylark |
| 2 | 18.0 | 8 | 318.0 | ... | 70 | 1 | plymouth satell |
| 3 | 16.0 | 8 | 304.0 | ... | 70 | 1 | amc rebel |
| 4 | 17.0 | 8 | 302.0 | ... | 70 | 1 | ford tor |

[5 rows x 9 columns]

# 머신러닝 회귀분석

```
df.info()
<class ‘pandas.core.frame.DataFrame’>

df.describe()
```