강의계획표

주	해당 장	주제
1	1장 2장, 3장	머신러닝이란
2		머신러닝을 위한 기초지식, 구현을 위한 도구
3	4장	선형 회귀로 이해하는 지도학습
4	5장	분류와 군집화로 이해하는 지도 학습과 비지도 학습
5	6장	다양한 머신러닝 기법들 다항 회귀, 결정 트리, SVM
6		
7	7장	인공 신경망 기초 - 문제와 돌파구
8		중간고사
9	8장	고급 인공 신경망 구현
10	9장	신경망 부흥의 시작, 합성곱 신경망
11	10장	순환 신경망
12	11장	차원축소와 매니폴드 학습
13	12장	오토인코더와 잠재표현 학습
14	13장	인공지능의 현재와 미래
15		보강주
16		기말고사

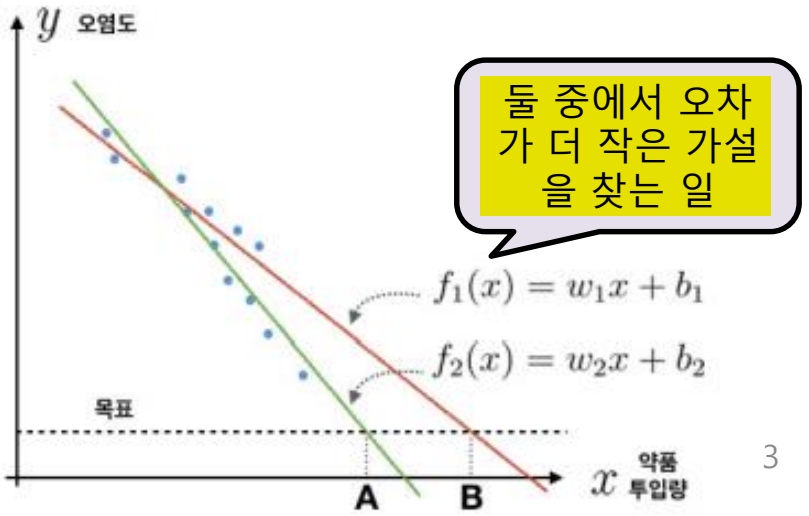
4장 지도학습-선형회귀

- 회귀 분석이란 무엇인가.
- 회귀 분석은 머신러닝의 범주에 들어가는가.
- 회귀 분석은 어떻게 동작하며 구현 방법은 어떤 것이 있는가.
- 학습에 사용되는 데이터는 어떻게 정제할 수 있는가.
- Scikit-learn 라이브러리를 이용한 기계학습

1. 선형회귀

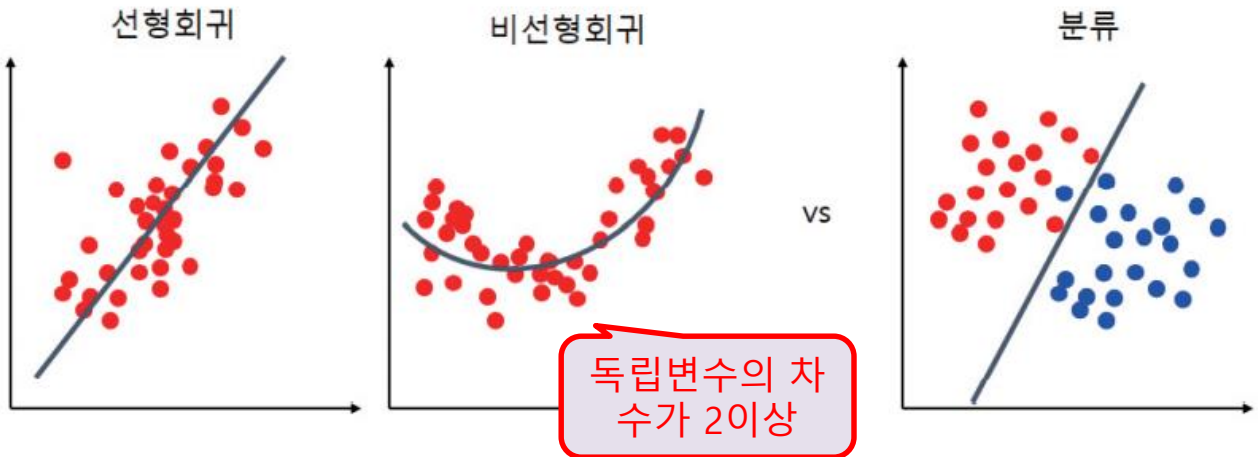
- ❖ 회귀regression :
  - 통계학에서 처음 사용
  - 회귀분석: 관측된 데이터를 통해 독립변수independent variable와 종속변수 dependent variable 사이의 숨어있는 관계를 추정하는 일
- ❖ 선형 회귀linear regression
  - $y = f(x)$에서 입력 x에 대응되는 실수 y들이 주어지고 추정치 $f(x)$가 가진 오차를 측정하고,
  - 이 오차를 줄이는 방향으로 함수의 계수(coefficient)를 **최적화**하는 일
  - 성능 척도 P는 예측한 값 $\hat{y}$과 데이터로 제공되는 목표값 y의 차이가 적을수록 높은 점수를 부여함

  - 약품투입량과 오염도의 관계
  - 모델, 가설hypothesis : $f_a(x)$, $f_b(x)$
  - 오차가 가장 적은 계수(w, b)



2. 선형회귀와 지도학습

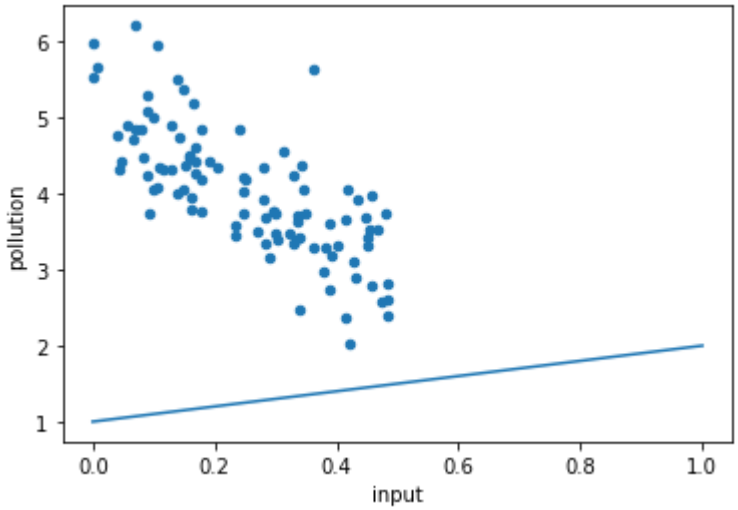
- ❖ 지도학습
  - 입력-출력Label 쌍을 **학습**한 후에 새로운 입력값이 들어왔을 때, 합리적인 출력값을 **예측**하는 것.
- ❖ 기계학습
  - 데이터 (x, y)를 주면 함수 $f(x)$를 만들어내는 일
  - **특징**feature : 관찰되는 현상에서 측정할 수 있는 개별적인 속성attribute
  - 분류classification : 입력 데이터를 이산적인 **범주**category 중의 하나로 대응
  - 회귀regression : 입력 데이터 하나 하나에 대응하는 출력값을 예측
    - 선형linear 회귀
    - 비선형nonlinear 회귀



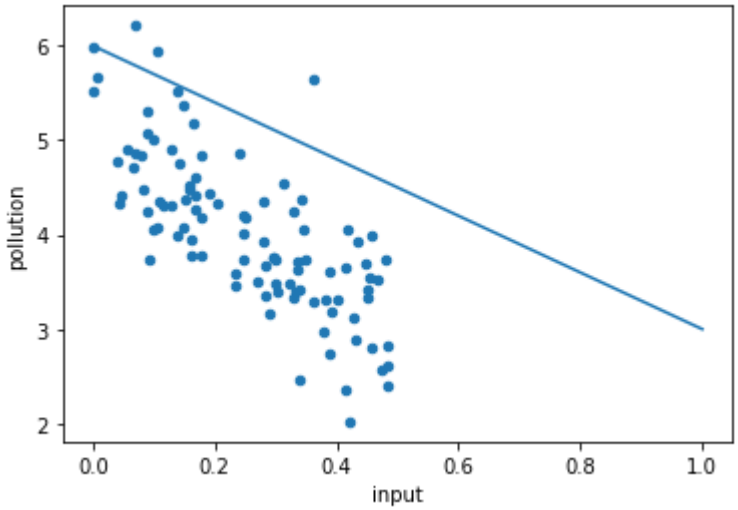
Python program

- 프로그램1
  - 데이터와 가설
  - https://colab.research.google.com/drive/1jB-Cnzgd30hzNPhIggXN-s8QG8dgsUUc
  - 가설: $y = wx + b$

$w = 1, b = 1$



$w = -3, b = 6$



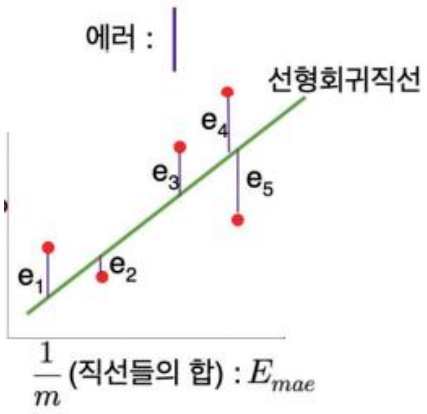
3. 모델의 오차

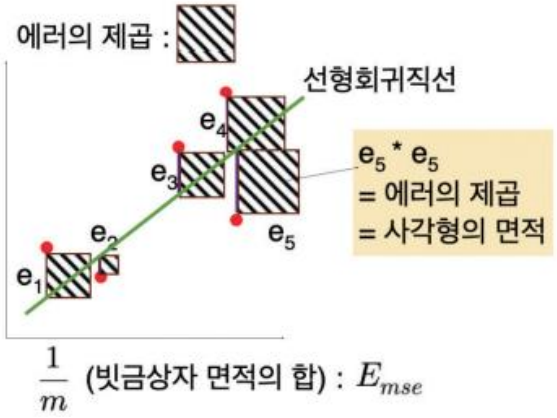
❖ 오차 척도

- 가설의 정확성을 평가하는 방법
- 평균 절대 오차mean absolute error:MAE
- 평균 제곱 오차mean square error:MSE

$$E_{mae} = \frac{1}{m}\sum_{i=1}^{m}|\hat{y}_i - y_i|$$

$$E_{mse} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

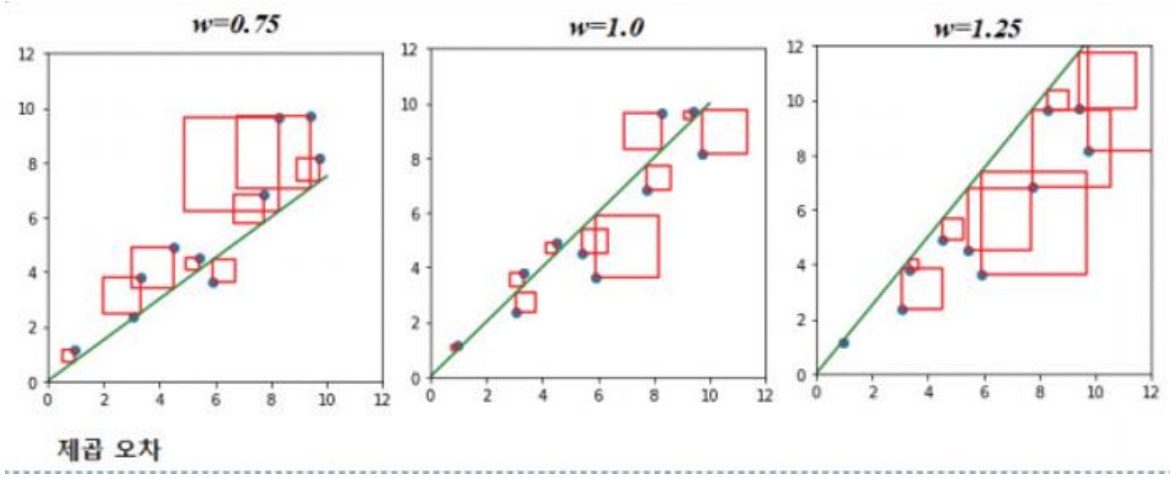


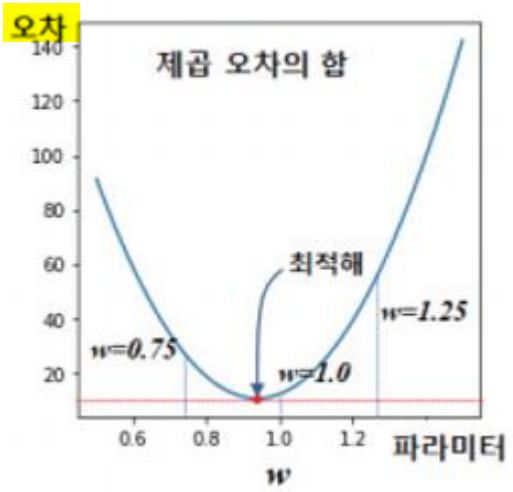


3. 모델의 오차

❖ 평균 제곱 오차

- 오차합 곡면의 기울기를 따라 내려가 최소 오차에 접근 (미분가능)
- 경사하강법gradient descent: 오차 기울기의 반대방향(오차가 줄어드는 방향)





4. 오차를 이용한 좋은 가설 찾기

❖ 좋은 가설

- 추정한 종속변수와 실제 종속변수의 차이가 적을수록 좋은 가설

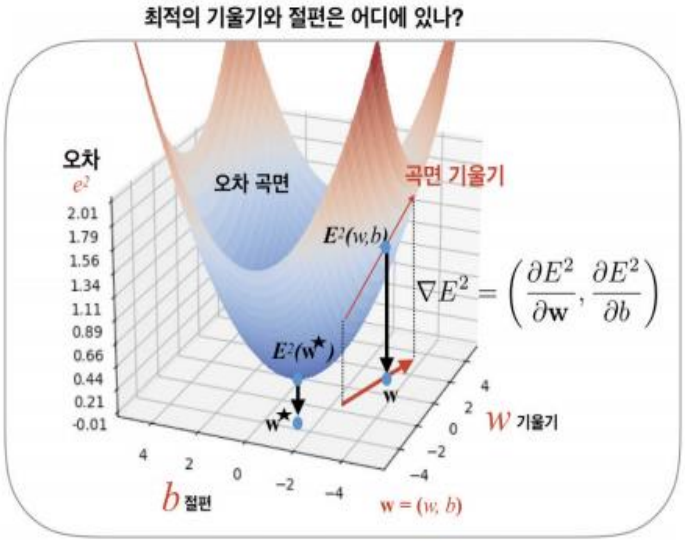
$$E = \hat{y} - y = wx + b - y$$

- 최소 제곱법least squares approximation: 오차를 제곱하여 오차 곡면의 기울기를 따라 내려가 기울기가 0인 극소 지점을 찾는 것

$$\nabla E^2 = \left(\frac{\partial E^2}{\partial w}, \frac{\partial E^2}{\partial b} \right)$$

$$\frac{\partial E^2}{\partial w} = \frac{\partial (wx + b - y)^2}{\partial w} = 2(wx + b - y)x = 2Ex$$

$$\frac{\partial E^2}{\partial b} = \frac{\partial (wx + b - y)^2}{\partial b} = 2(wx + b - y) \cdot 1 = 2E$$



- 가중치$(\boldsymbol{w}, \boldsymbol{b})$ **학습**

$$w \leftarrow w - \eta \sum_{i=1}^{n} E_i x_i, \quad b \leftarrow b - \eta \sum_{i=1}^{n} E_i$$

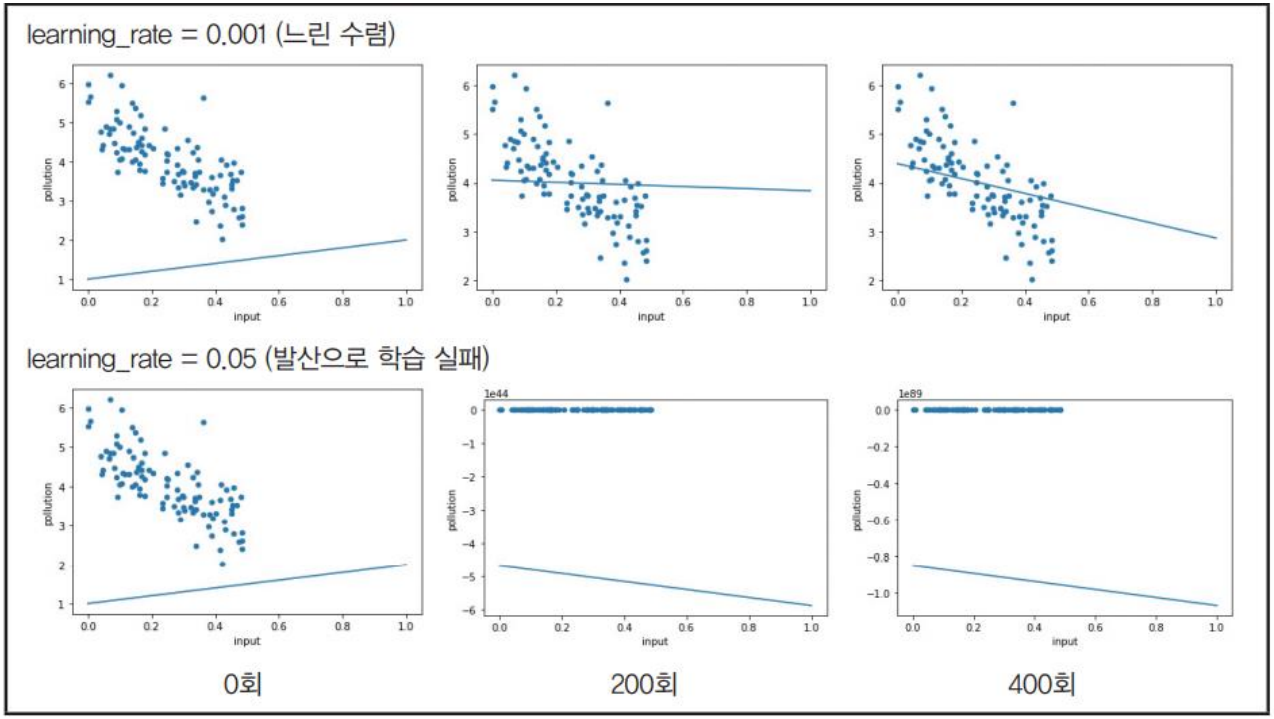
5. 기계 학습의 개념으로 해석하는 선형 회귀

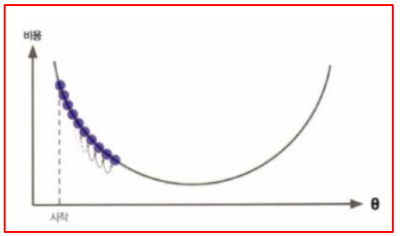
❖ 선형회귀

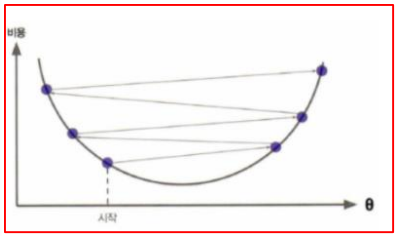
- 모델: 선형 방정식
- Parameter: 직선의 기울기와 절편 (w, b) 학습

$$w \leftarrow w - \eta \sum_{i=1}^{n} E_i x_i, \quad b \leftarrow b - \eta \sum_{i=1}^{n} E_i$$

- Hyper parameter: 학습률learning rate, 학습반복 횟수learning iteration



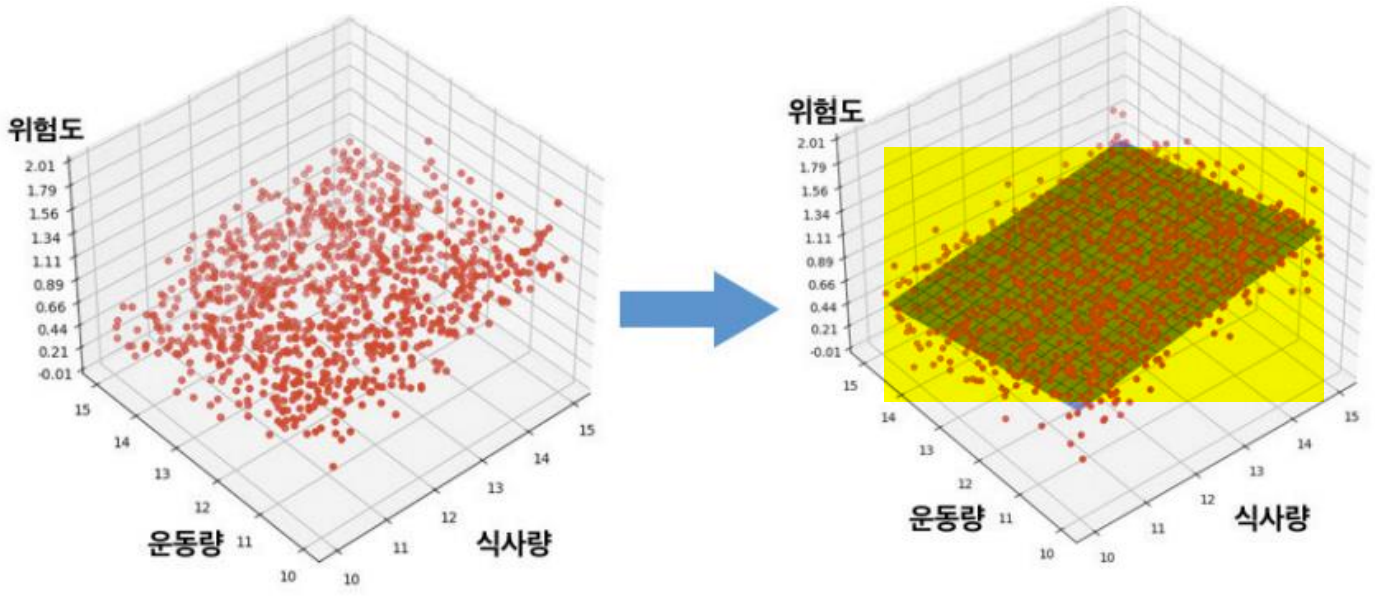




6. 다변량 회귀

- 다변량multivariate 선형회귀
  - 입력에 사용되는 특징이 n개
  - 예) 식사량과 운동량에 따른 건강위험도

$$\hat{y} = w_1 x_1 + w_2 x_2 + b = \mathbf{w}^{\mathrm{T}} \mathbf{x} + b$$



6. 다변량 회귀

- 다변량multivariate 선형회귀
  - n차원 공간의 초평면hyperplane

$$\hat{y} = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

$$\hat{y} = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \theta^T \mathbf{x}$$

  - Instance수가 m, i 번째 instance $x^{(i)}$

$$X^{(i)} = (1, x_1^{(i)}, x_2^{(i)}, \cdots, x_n^{(i)})^T$$

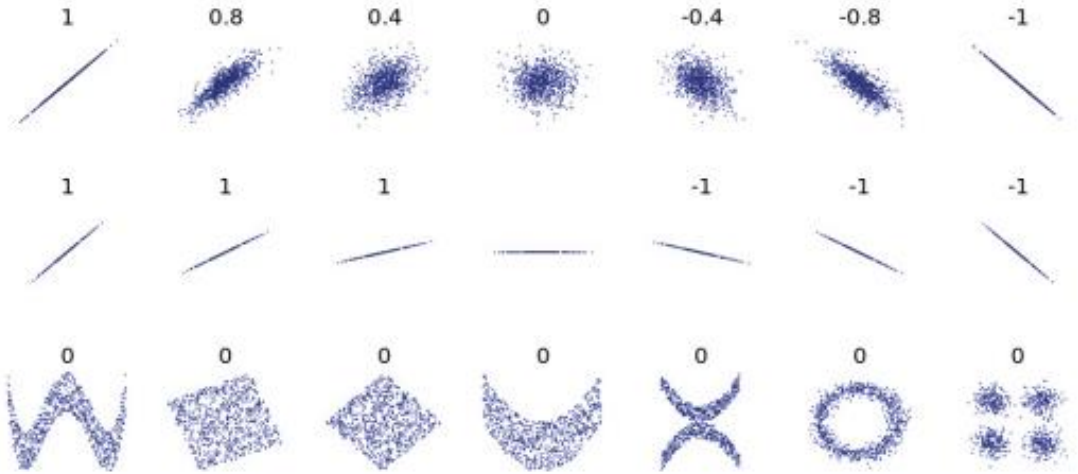
  - 평균제곱오차 $E_{mse}(\mathbf{x}, \theta) = \frac{1}{m}\sum_{i=1}^{n}(\theta^T \mathbf{x}^{(i)} - y^{(i)})^2$ 가 최소가 되는 θ를 찾음

$$\frac{\partial E_{mse}(\mathbf{x}, \theta)}{\partial \theta_j} = \frac{2}{m}\sum_{i=1}^{n}(\theta^T \mathbf{x}^{(i)} - y^{(i)})x_j^{(i)}$$

$$\theta \leftarrow \theta - \eta \nabla_\theta E_{mse}$$

7. 다변량 데이터 특징들 사이의 상관관계

- https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who
- 데이터 크기와 결손값 확인: shape(), isnull()
- 상관행렬correlation matrix
  - 데이터 속성(자질)간의 상관관계 파악: corr()
  - Pearson 상관계수(위키참조)



$$\text{피어슨상관계수} = \frac{\text{공분산}}{\text{표준편차} \cdot \text{표준편차}}$$

$$r_{XY} = \frac{\frac{\sum_i^n (X_i - \overline{X})(Y_i - \overline{Y})}{n-1}}{\sqrt{\frac{\sum_i^n (X_i - \overline{X})^2}{n-1}} \sqrt{\frac{\sum_i^n (Y_i - \overline{Y})^2}{n-1}}}$$

따라서

$$r_{XY} = \frac{\sum_i^n (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_i^n (X_i - \overline{X})^2} \sqrt{\sum_i^n (Y_i - \overline{Y})^2}}$$

- 상관관계 도식화
  - seaborn 라이브러리의 heatmap() 함수
  - pairplot()

Python program

❖ 성능개선

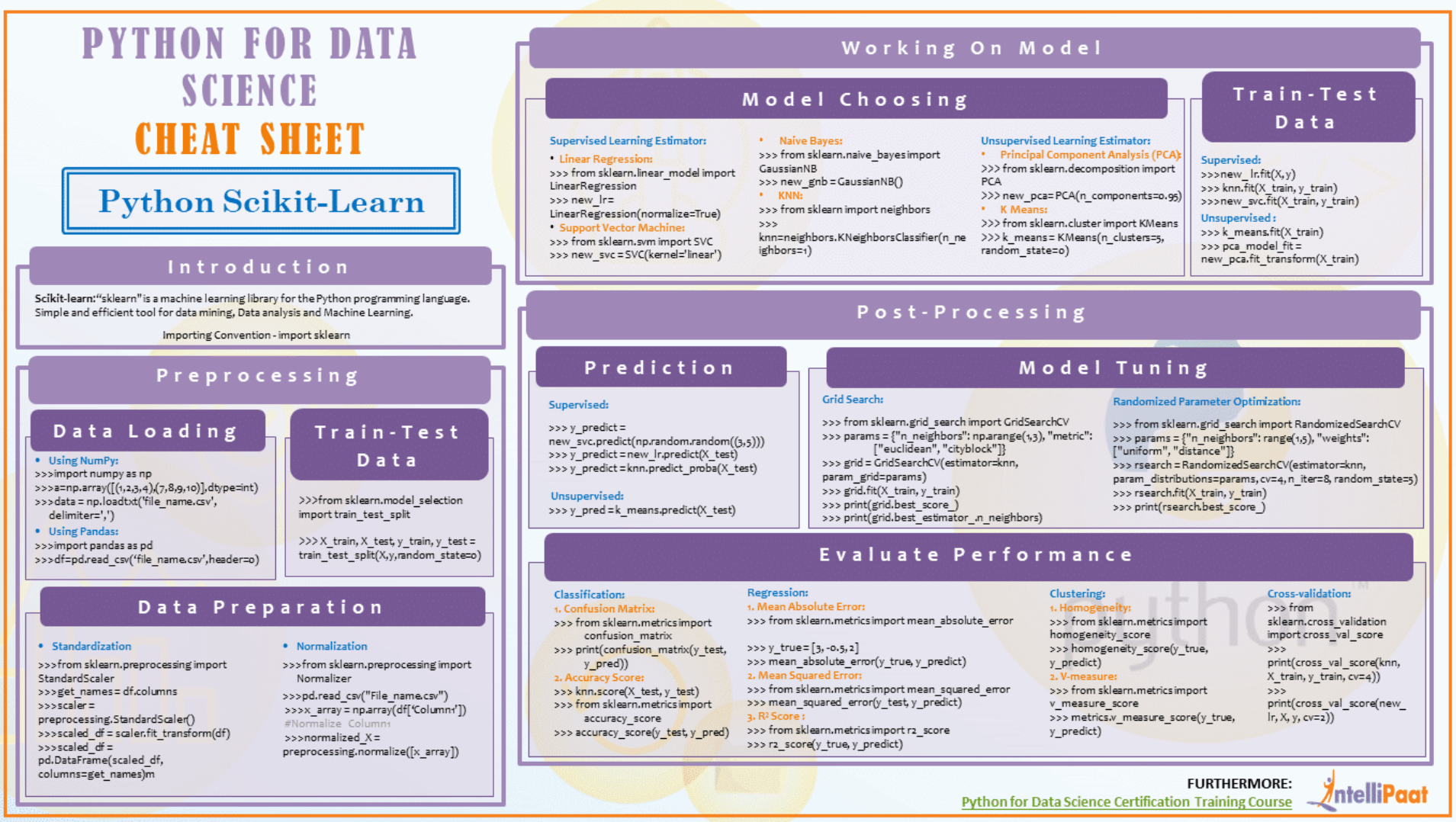
- 데이터 분리(훈련용, 검증용, 테스트용)

- 데이터의 정규화normalization $\tilde{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$

- 데이터의 표준화standardization $x' = \frac{x - \mu_x}{\sigma_x}$

- https://colab.research.google.com/drive/1jB-Cnzgd30hzNPhIggXN-s8QG8dgsUUc

Scikit-Learn (3)



PYTHON FOR DATA SCIENCE
CHEAT SHEET

Python Scikit-Learn

Introduction

Scikit-learn: "sklearn" is a machine learning library for the Python programming language. Simple and efficient tool for data mining, Data analysis and Machine Learning.

Importing Convention - import sklearn

Preprocessing

Data Loading

- Using NumPy:

```
>>>import numpy as np
>>>a=np.array([(1,2,3,4)(7,8,9,10)],dtype=int)
>>>data = np.loadtxt('file_name.csv',
   delimiter=',')
```

- Using Pandas:

```
>>>import pandas as pd
>>>df=pd.read_csv('file_name.csv',header=0)
```

Train-Test Data

```
>>>from sklearn.model_selection
import train_test_split

>>> X_train, X_test, y_train, y_test =
train_test_split(X,y,random_state=0)
```

Data Preparation

- Standardization

```
>>>from sklearn.preprocessing import
StandardScaler
>>>get_names= df.columns
>>>scaler =
preprocessing.StandardScaler()
>>>scaled_df = scaler.fit_transform(df)
>>>scaled_df =
pd.DataFrame(scaled_df,
columns=get_names)m
```

- Normalization

```
>>>from sklearn.preprocessing import
   Normalizer
>>>pd.read_csv("File_name.csv")
>>>x_array = np.array(df['Column1'])
#Normalize  Column1
>>>normalized_X =
preprocessing.normalize([x_array])
```

Working On Model

Model Choosing

Supervised Learning Estimator:

- Linear Regression:

```
>>> from sklearn.linear_model import
LinearRegression
>>> new_lr=
LinearRegression(normalize=True)
```

- Support Vector Machine:

```
>>> from sklearn.svm import SVC
>>> new_svc = SVC(kernel='linear')
```

- Naive Bayes:

```
>>> from sklearn.naive_bayes import
GaussianNB
>>> new_gnb = GaussianNB()
```

- KNN:

```
>>> from sklearn import neighbors
>>>
knn=neighbors.KNeighborsClassifier(n_ne
ighbors=1)
```

Unsupervised Learning Estimator:

- Principal Component Analysis (PCA):

```
>>> from sklearn.decomposition import
PCA
>>> new_pca= PCA(n_components=0.95)
```

- K Means:

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=5,
random_state=0)
```

Train-Test Data

Supervised:

```
>>>new_ lr.fit(X, y)
>>> knn.fit(X_train, y_train)
>>>new_svc.fit(X_train, y_train)
```

Unsupervised :

```
>>> k_means.fit(X_train)
>>> pca_model_fit =
new_pca.fit_transform(X_train)
```

Post-Processing

Prediction

Supervised:

```
>>> y_predict =
new_svc.predict(np.random.random((5,5)))
>>> y_predict = new_lr.predict(X_test)
>>> y_predict = knn.predict_proba(X_test)
```

Unsupervised:

```
>>> y_pred = k_means.predict(X_test)
```

Model Tuning

Grid Search:

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3), "metric":
        ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

Randomized Parameter Optimization:

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": range(1,5), "weights":
["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
param_distributions=params, cv=4, n_iter=8, random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

Evaluate Performance

Classification:

1. Confusion Matrix:

```
>>> from sklearn.metrics import
      confusion_matrix
>>> print(confusion_matrix(y_test,
      y_pred))
```

2. Accuracy Score:

```
>>> knn.score(X_test, y_test)
>>> from sklearn.metrics import
      accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Regression:

1. Mean Absolute Error:

```
>>> from sklearn.metrics import mean_absolute_error

>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_predict)
```

2. Mean Squared Error:

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_predict)
```

3. R^2 Score :

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_predict)
```

Clustering:

1. Homogeneity:

```
>>> from sklearn.metrics import
homogeneity_score
>>> homogeneity_score(y_true,
y_predict)
```

2. V-measure:

```
>>> from sklearn.metrics import
v_measure_score
>>> metrics.v_measure_score(y_true,
y_predict)
```

Cross-validation:

```
>>> from
sklearn.cross_validation
import cross_val_score
>>>
print(cross_val_score(knn,
X_train, y_train, cv=4))
>>>
print(cross_val_score(new_
lr, X, y, cv=2))
```

FURTHERMORE:
Python for Data Science Certification Training Course

IntelliPaat