생산공정 모니터링을 위한 데이터 사이언스 입문

홍익대학교
기계·시스템디자인공학과
Biomechanics Research Laboratory

오 유 근 교수

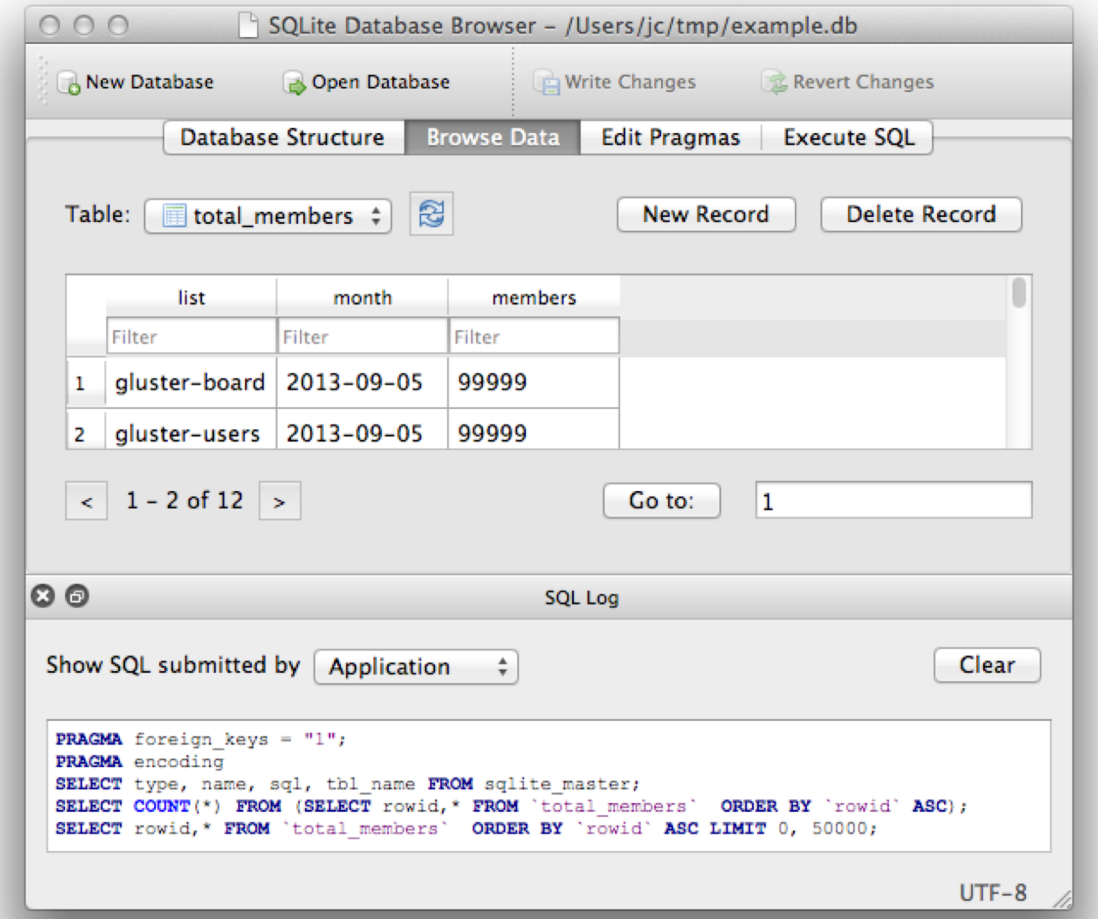
Ch3. Database Programming

SQL (Structured Query Language)

- RDBMS
  - Relational Database Management System
  - Ex: Oracle, MS SQL Server, Microsoft Access, MySQL …
- SQL
  - Structured Query Language
  - RDBMS에서 data 조회, insert, update, delete
- Table
  - RDBMS에서 data를 저장하는 object
  - Primary key(s)
  - Row, column

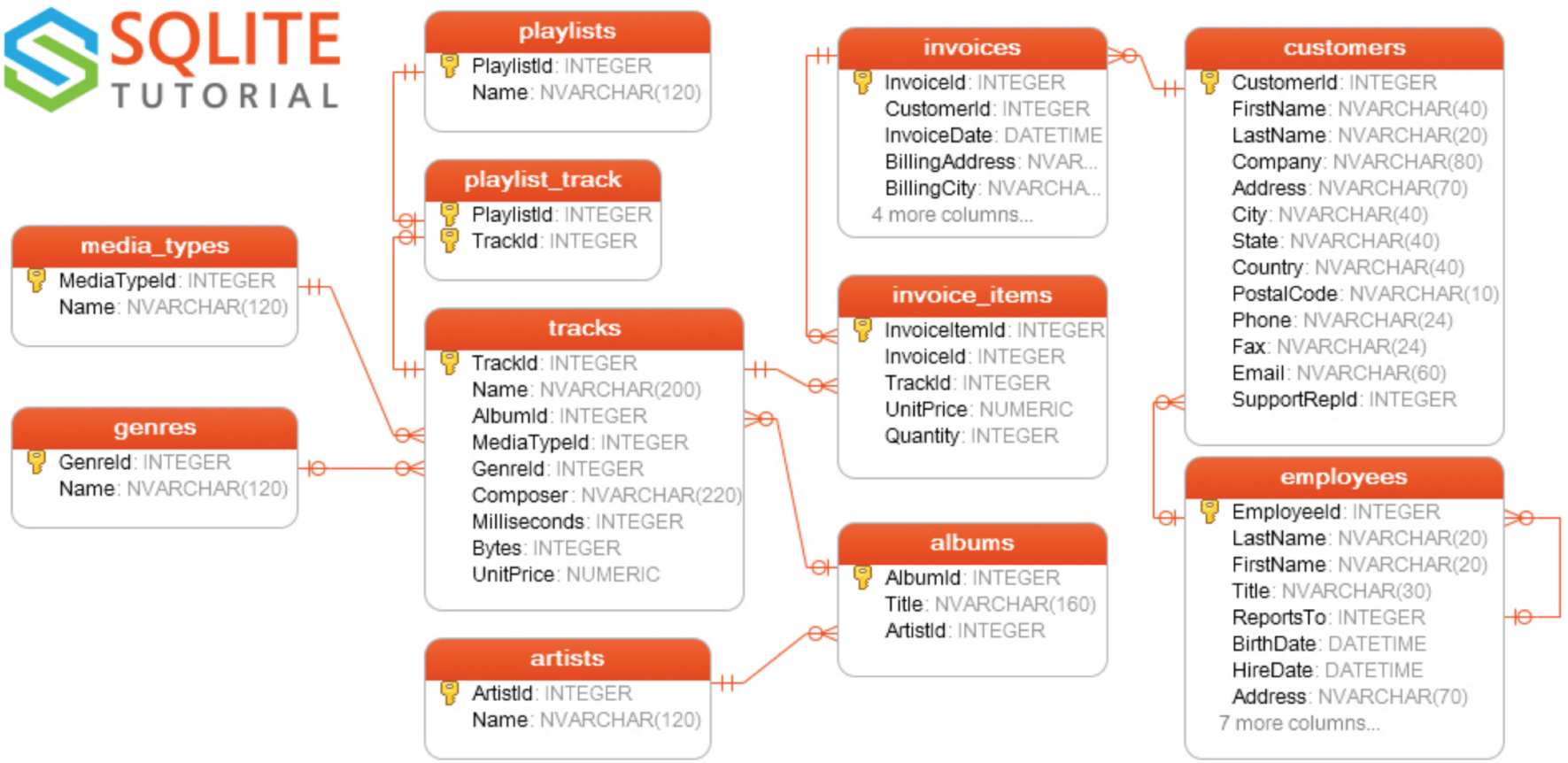
Weather			
city	**state**	**high**	**low**
Phoenix	Arizona	105	90
Tucson	Arizona	101	92
Flagstaff	Arizona	88	69
San Diego	California	77	60
Albuquerque	New Mexico	80	72

SQLite 설치



https://sqlitebrowser.org/

Sample Database



https://www.sqlitetutorial.net/sqlite-sample-database/

SQL 문법 (SELECT ~ FROM ~)

- Data 조회
- SELECT *
  FROM table_name
- SELECT col1, col2, col3
  FROM table_name
- SELECT col1 as co1_alias, col2 as co2_alias
  FROM table_name
- SELECT DISTINCT col1, col2, col3
  FROM table_name

SQL 문법 (WHERE)

- 검색조건
- Ex) SELECT *
  FROM table_name
  WHERE col1 = 1 and col2 ='Mexico' (conditions)

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

SQL 문법 (ORDER BY)

- 정렬을 위한 keyword
- 오름차순(ASC), 내림차순(DESC)
- Ex) SELECT col1, col2
  FROM table_name
  ORDER BY col1 ASC, col2 DESC

SQL 문법 (GROUP BY)

- Data grouping
- Ex) SELECT col1, col2
  FROM table_name
  GROUP BY col1 ASC, col2 DESC
- Count, Min, Max, Avg, Sum 등의 함수와 같이 사용

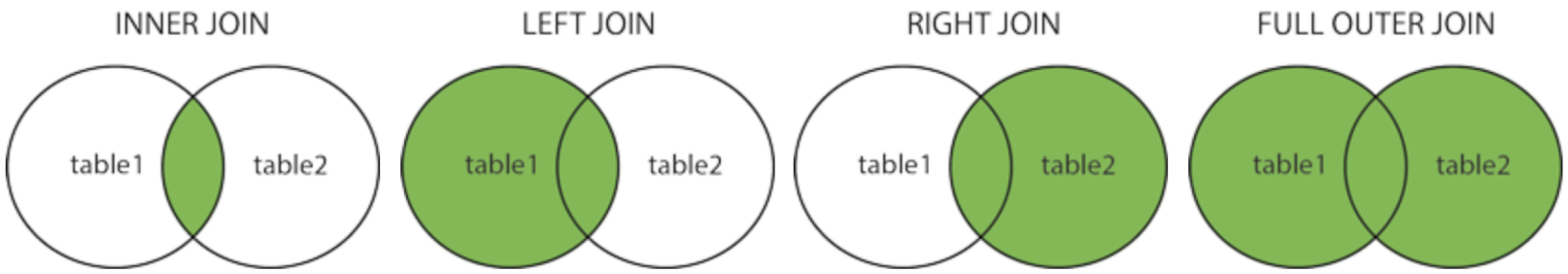
SQL 문법 (함수)

- MIN(), MAX()
  - SELECT MIN(col1) as Min_Col1, MAX(col2) as Max_Col2
    FROM table_name
    WHERE condition

- COUNT(), AVG(), SUM()
  - SELECT COUNT(col1), AVG(col2), SUM(col2)
    FROM table_name
    WHERE condition

SQL 문법 (Join)

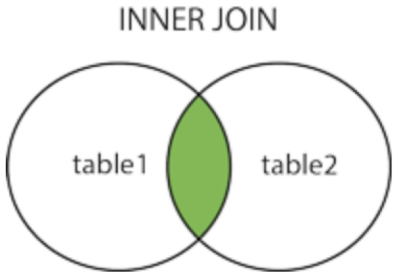
- 교집합, 차집합, 합집합



- INNER JOIN (교집합)
  - SELECT col_name1, col_name2
    FROM table1
    INNER JOIN table2
    ON table1.col_name3 = table2.col_name3

SQL 문법 (JOIN)

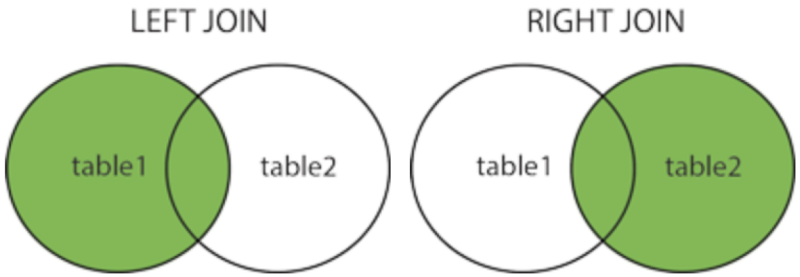
- INNER JOIN (교집합)



- INNER JOIN (교집합)
  - SELECT Orders.Order_ID, Customers.CustomerName
    FROM Orders
    INNER JOIN Customers
    ON Orders.col_CustomerID = Customers.CustomerID

SQL 문법 (JOIN)

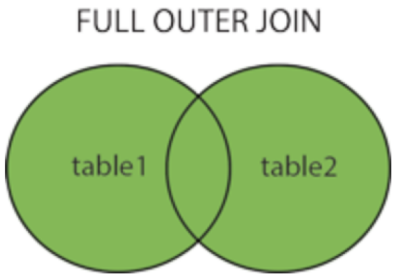
- LEFT/RIGHT JOIN (차집합)



- LEFT/RIGHT JOIN (교집합)
  - SELECT Orders.Order_ID, Customers.CustomerName
    FROM Orders
    LEFT JOIN Customers
    ON Orders.col_CustomerID = Customers.CustomerID

SQL 문법 (JOIN)

- FULL JOIN (합집합)



- LEFT/RIGHT JOIN (교집합)
  - SELECT col1, col2
    FROM table1
    FULL OUTER JOIN table2
    ON table1.col_name = table2.col_name
    WHERE condition

SQL 문법 (UNION)

- 두개 이상의 SELECT 구문을 합칠 때 사용
  - SELECT A.col1, A.col2 FROM table1 as A
    UNION
    SELECT B.col1, B.col2 FROM table2 as B

  - SELECT C.col1, C.col2
    FROM (SELECT A.col1, A.col2 FROM table1 as A
    UNION
    SELECT B.col1, B.col2 FROM table2 as B) as C
    WHERE conditions

SQL 문법 (INSERT/UPDATE/DELETE)

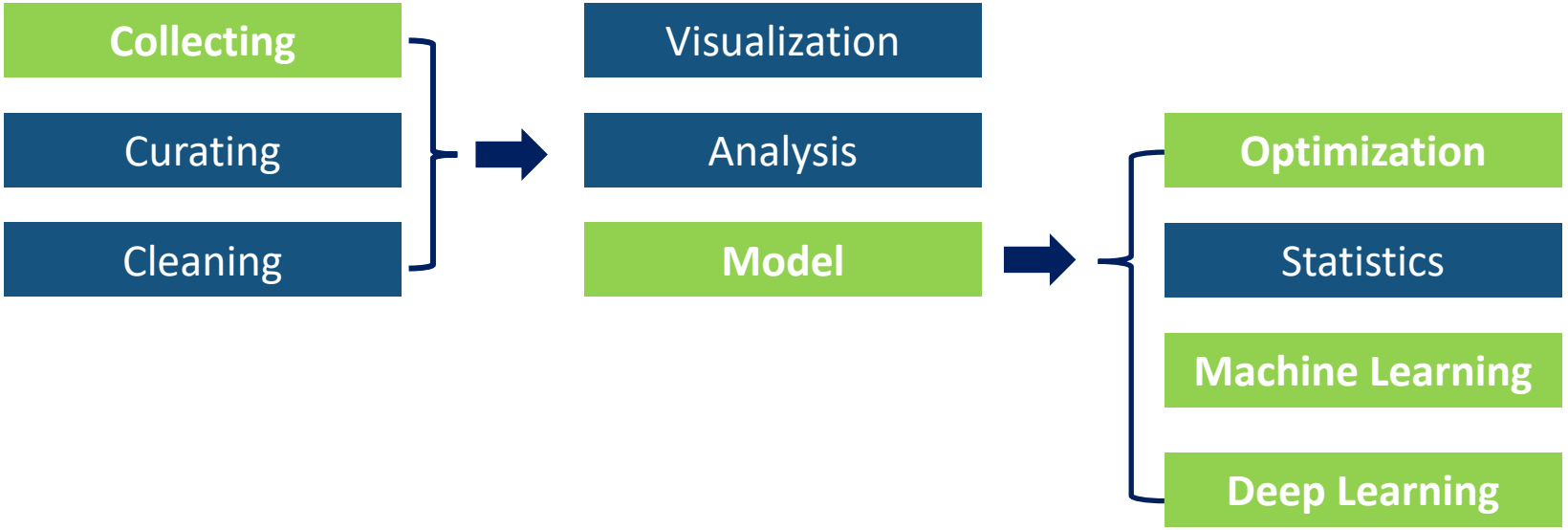
- INSERT INTO table_name (col1, col2, col3, …)
  Values (values, value2, value3, …)

- UPDATE table_name
  SET col1 = new_value1, col2 = new_value2, …
- WHERE condition

- DELETE FROM table_name
  WHERE condition

Ch4. Machine Learning

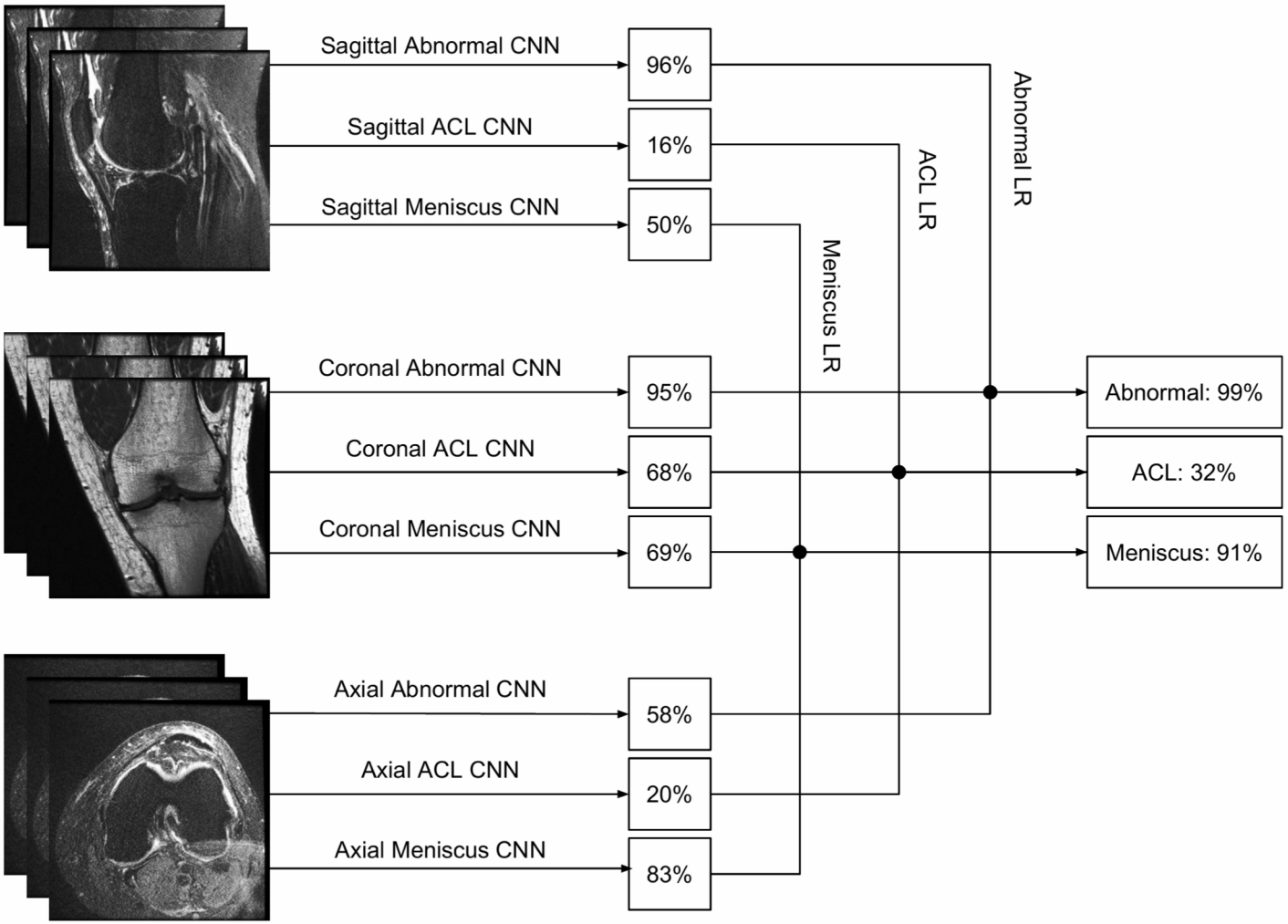
Data Science



금값 예측?

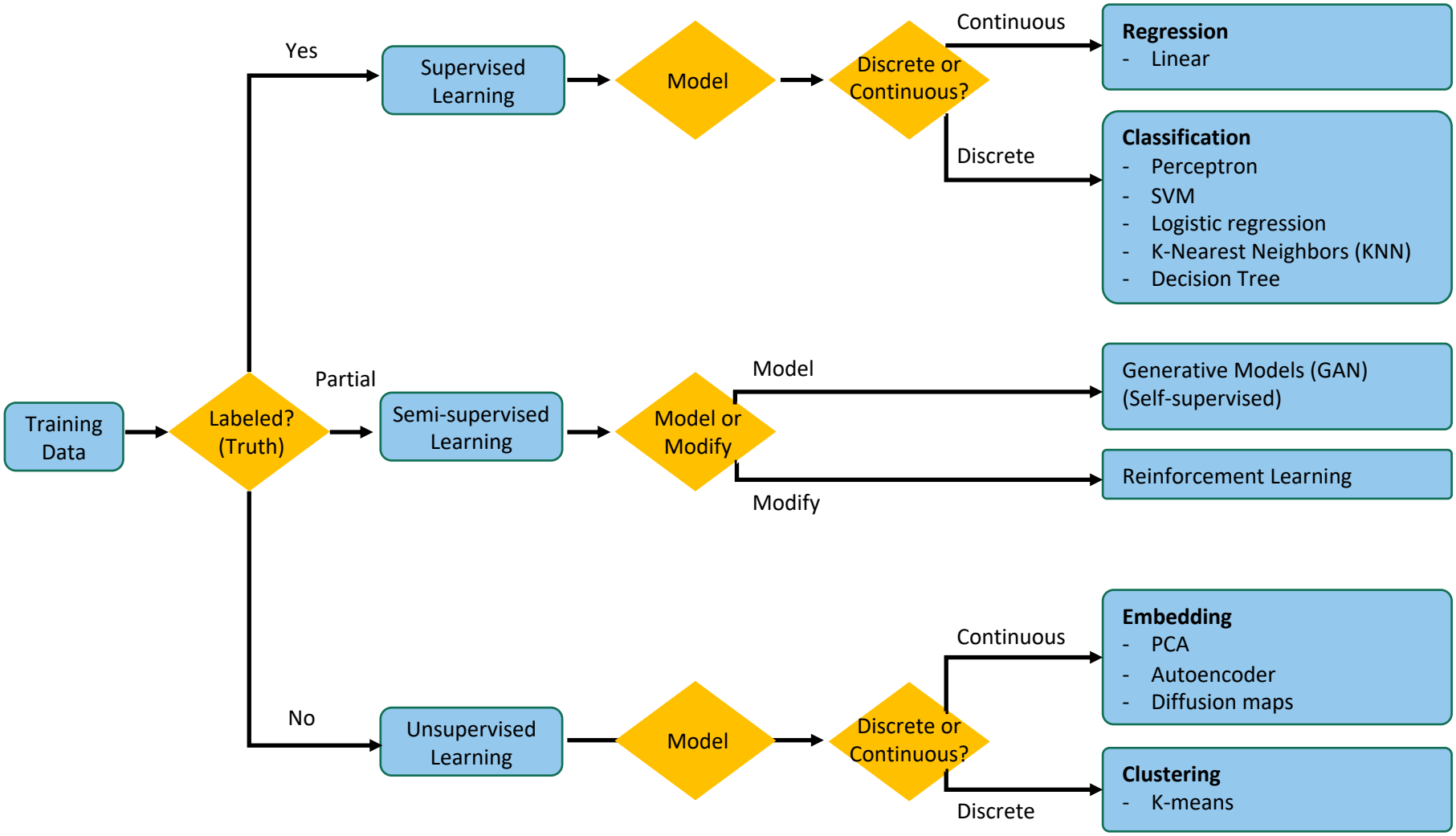

국제 금 선물(20-02) 금시세
1,568.60 달러/트로이온스
전일대비 ▲3.50 (+0.22%)
2020.02.07 COMEX(뉴욕상품거래소) 기준
1개월 3개월 1년 3년 5년 10년
최고 1,583.50 (01/30)
최저 1,452.10 (11/12)
1,604.62
1,577.01
1,549.40
1,521.79
1,494.18
1,466.57
1,438.96
11/11 11/15 11/22 11/29 12/06 12/13 12/20 12/27 01/03 01/10 01/17 01/24 01/31
?

무릎 부상 여부?



https://stanfordmlgroup.github.io/projects/mrnet/

Machine Learning

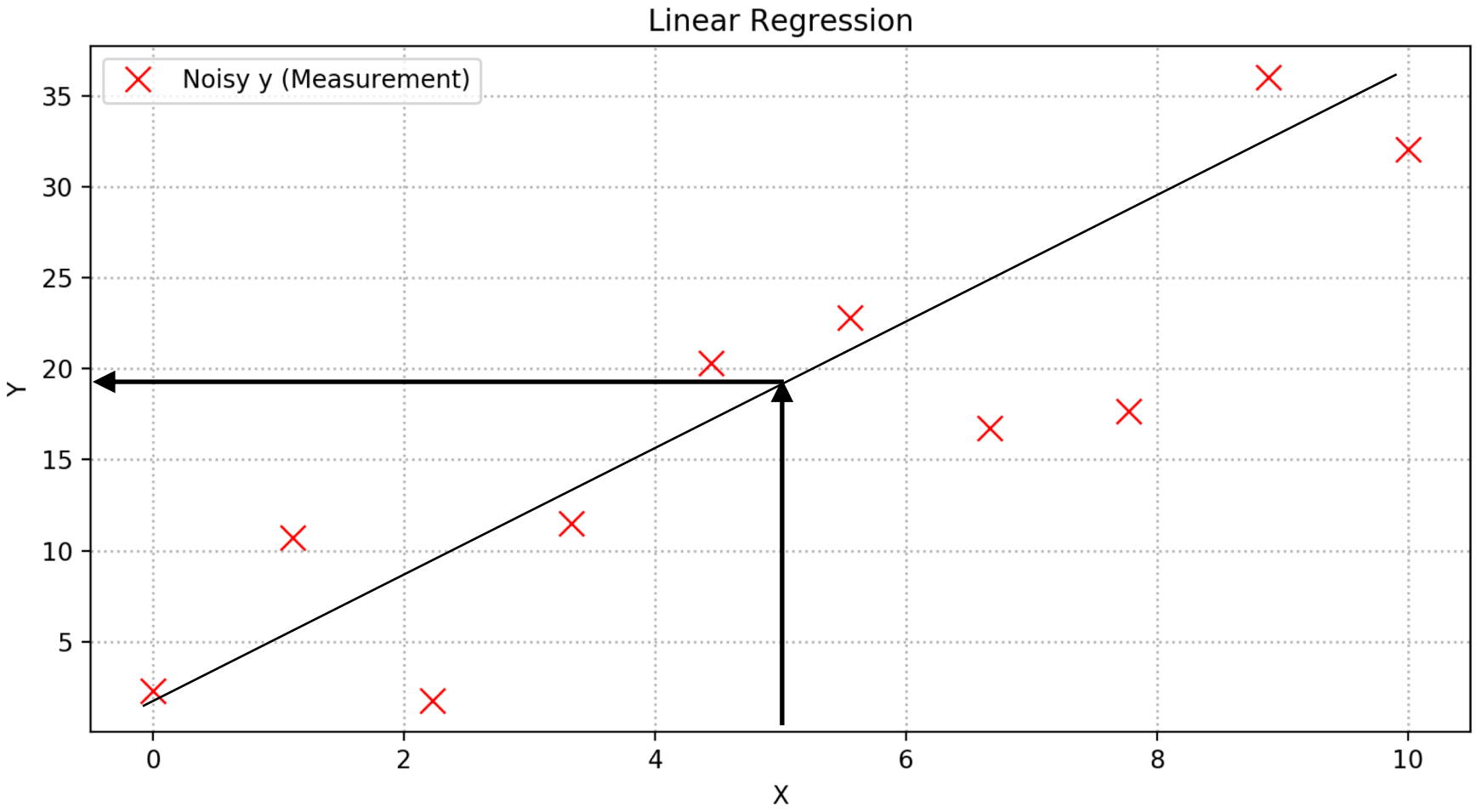


Adapted from Lecture by Prof. Steven Brunton
@ University of Washington
https://www.youtube.com/watch?v=0_IKUPYEYyY

Supervised Learning

Regression

Regression

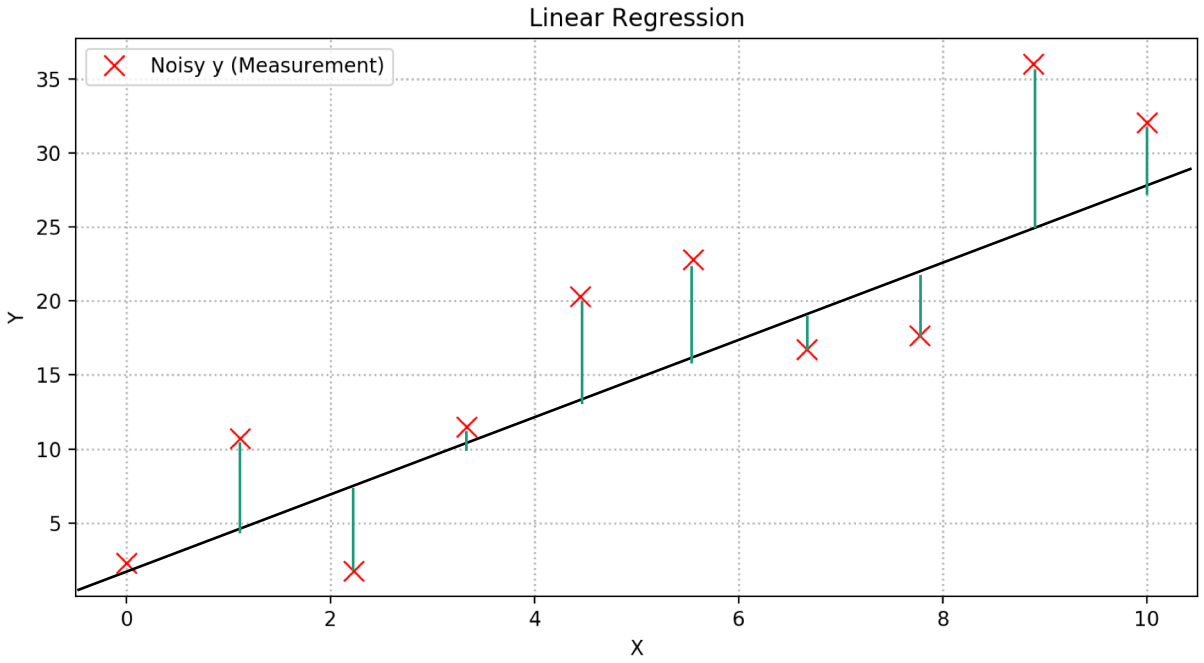

Linear Regression
Noisy y (Measurement)
35
30
25
20
15
10
5
Y
0
2
4
6
8
10
X

[실습] Linear Regression w/ Noisy Data

- Step 1
  - np.linspace를 이용하여 X vector 생성하기, $X \in [0, 10]$
  - $Y_{Ideal} = 3*X + 2$
  - $Y_{Noisy} = Y_{Ideal} + Noise$
  - Noise = np.random.normal(mu, sigma, X.size)

- Step 2
  - Y_{Ideal} 와 Y_{Noisy} 를 그래프 표현하기
  - https://matplotlib.org/tutorials/introductory/sample_plots.html

Optimization Problem



$$\text{Error} = \sum_{i=1}^{N} (\hat{y}(x_i) - y_i)^2 = \sum_{i=1}^{N} (a * x_i + b) - y_i)^2 = \sum_{i=1}^{N} (\theta_1 a * x_i + \theta_0) - y_i)^2$$

$$\rightarrow \min_{\theta} \sum_{i=1}^{N} (\hat{y}(x_i) - y_i)^2$$

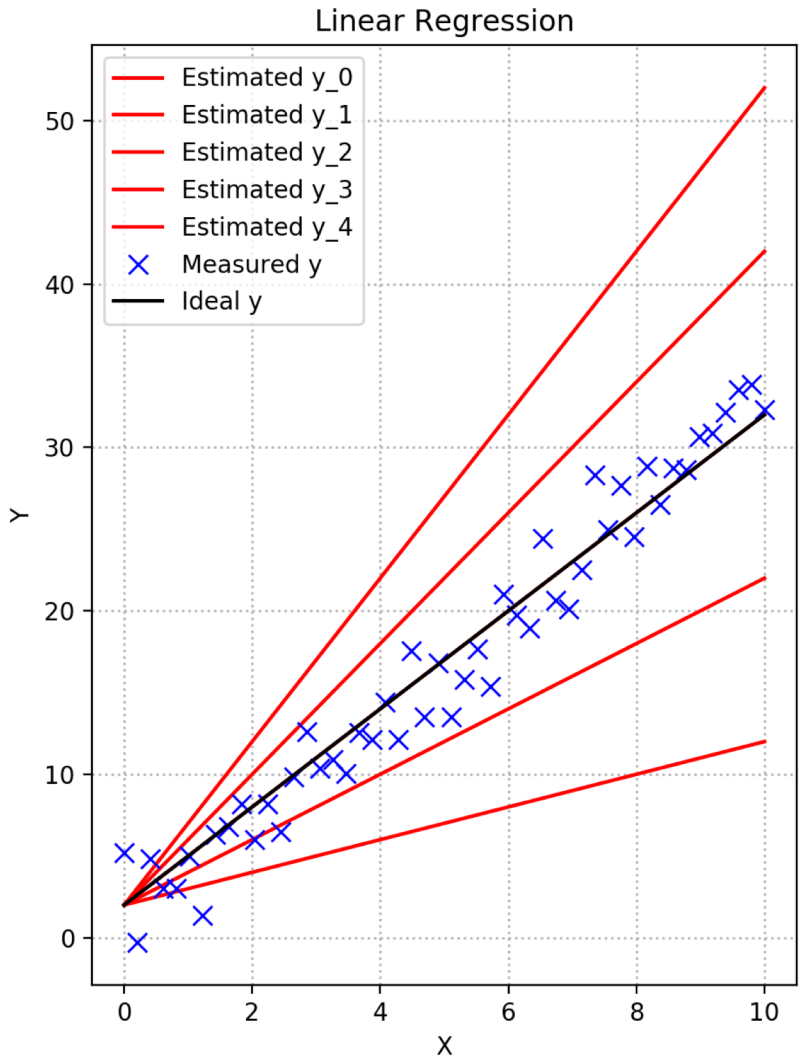
[실습] Linear Regression w/ Noisy Data

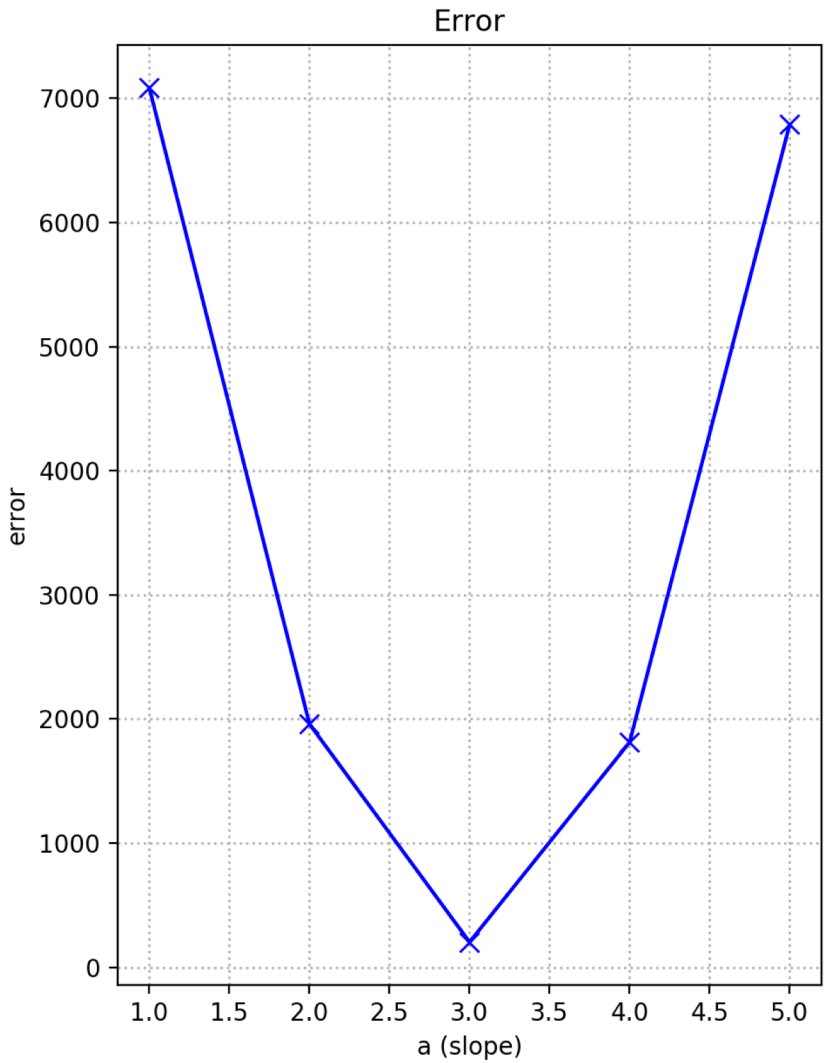
- Step 1
  - np.linspace를 이용하여 X vector 생성하기, $X \in [0, 10]$
  - $Y_{Ideal} = 3*X + 2$
  - $Y_{Noisy} = Y_{Ideal} + Noise$
  - Noise = np.random.normal(mu, sigma, X.size)

- Step 2
  - Y_{Ideal} 와 Y_{Noisy} 를 그래프 표현하기
  - https://matplotlib.org/tutorials/introductory/sample_plots.html

- Step 3
  - Error 함수를 정의하고 a=[1,2,3,4,5]로 변화시키며 Error의 변화 추이를 그래프로 표현

[실습] 기울기를 바꿔본다면 Error는?





Least Square Estimation

- A parameterized or model-based identification method to relate an observable variable $y(t)$ to p explanatory variables or **regressors**, $\emptyset_1(t)$, $\emptyset_2(t)$, $\ldots\emptyset_p(t)$.

$$y(t) = \emptyset_1(t)\vartheta_1 + \emptyset_2(t)\vartheta_2 + \cdots \emptyset_p(t)\vartheta_p + e(t)$$

- Let t = 1, … , N,

Observed variable ➔

$$y := [y(1), y(2), \ldots, y(N)]^T \quad : \quad (N \times 1)\ column\ vector$$

Unobserved error ➔

$$e := [e(1), e(2), \ldots, e(N)]^T \quad : \quad (N \times 1)\ column\ vector$$

Unknown parameters ➔

$$\vartheta := [\vartheta_1, \vartheta_2, \ldots, \vartheta_p]^T \quad : \quad (p \times 1) column\ vector$$

Observation matrix (regressor) ➔

$$\Phi : (N \times p)\ matrix\ with\ elements\ \emptyset_{t_j} = \emptyset_j(t), j = 1,2, \ldots, p$$

$$y = \Phi\vartheta + e$$

Least Square Estimation

- The unknown parameters can be estimated by minimizing the prediction error or residuals $\varepsilon(t)$.

$$\varepsilon(t) = y(t) - \Phi\vartheta$$

$$J(\vartheta) = \varepsilon^T\varepsilon = (y^T - \vartheta^T\Phi^T)(y - \Phi\vartheta)$$

- Then, the ***ordinary*** least-squares estimate is as below.

$$\hat{\vartheta} = (\Phi^T\Phi)^{-1}\Phi^T y$$

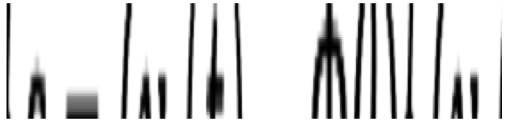
- $(\Phi^T\Phi)^{-1}\Phi^T$ is called the pseudo or generalized inverse of Φ.

Least Square Estimation

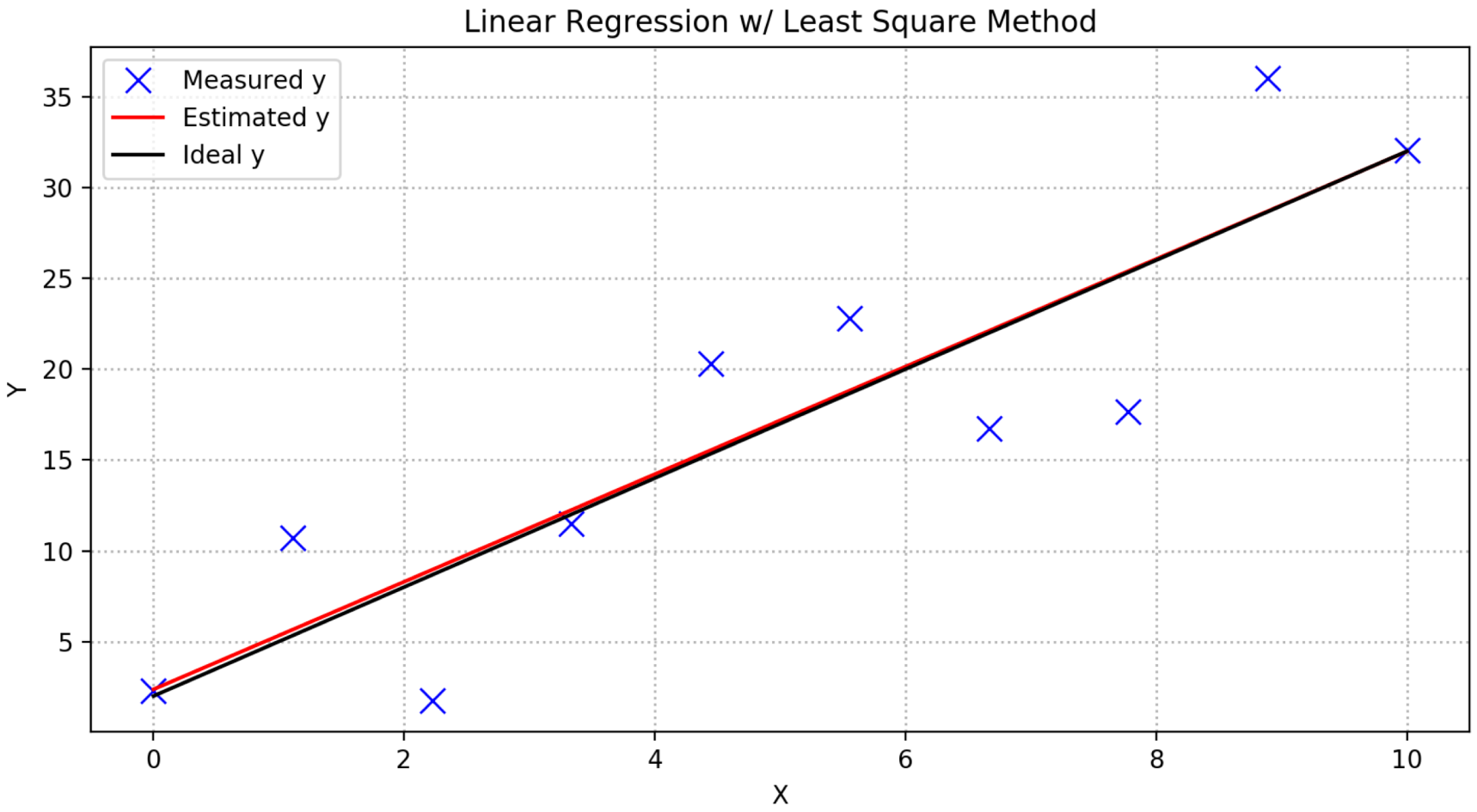
$$\text{Error} = \sum_{i=1}^{N} (\hat{y}(x_i) - y_i)^2 = \sum_{i=1}^{N} (a * x_i + b) - y_i)^2 = \sum_{i=1}^{N} (\theta_1 a * x_i + \theta_0) - y_i)^2$$

$$\varepsilon(t) = y_i(t) - \Phi\theta$$

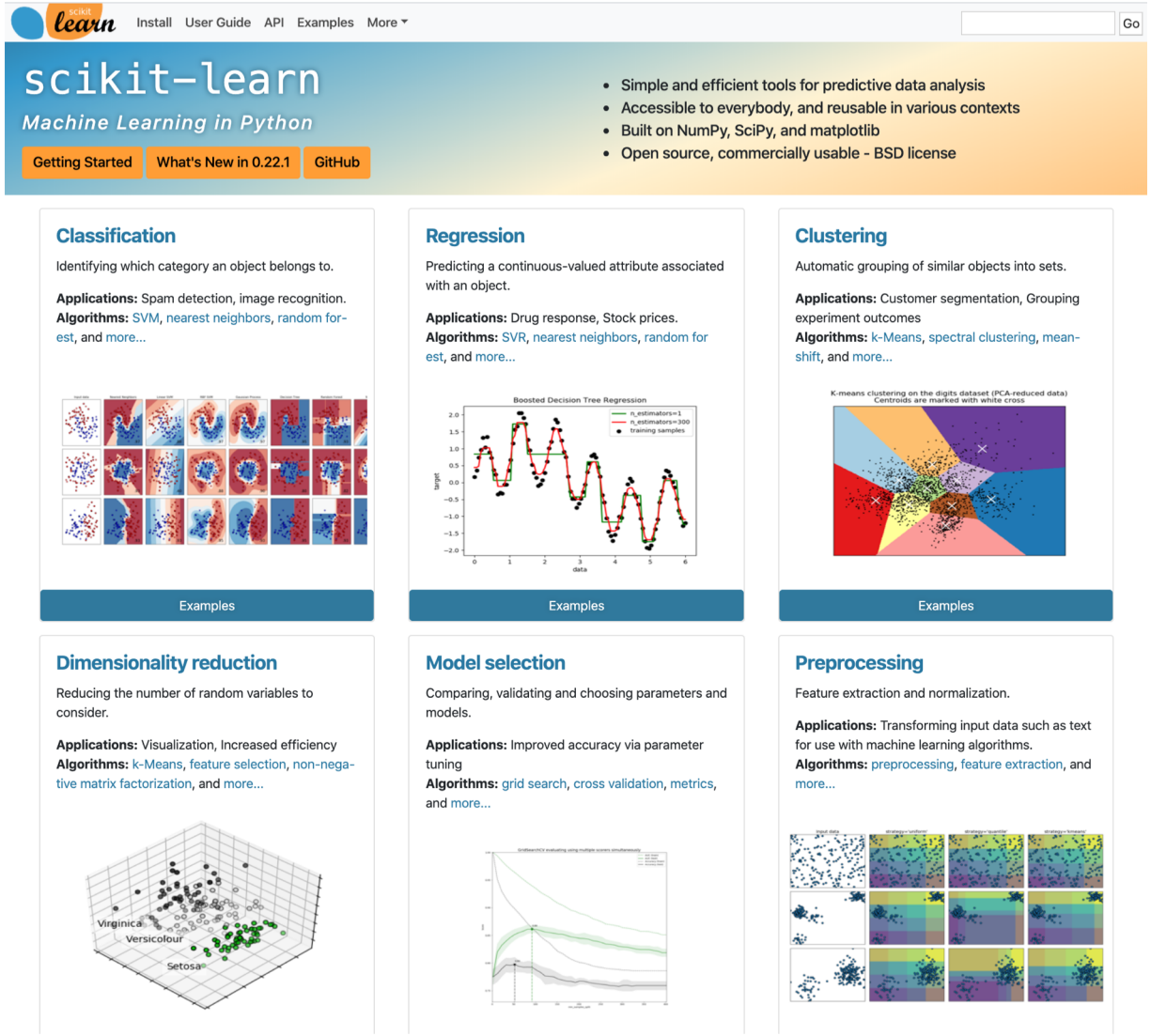
$$J(\theta) = \varepsilon^T \varepsilon = (y_i(t) - \Phi\theta)^T (y_i(t) - \Phi\theta)$$

- Jacobian (Error²) 은  일 때, 최소(극소)

[실습] Least Square Method 활용한 Regression



Scikit-Learn

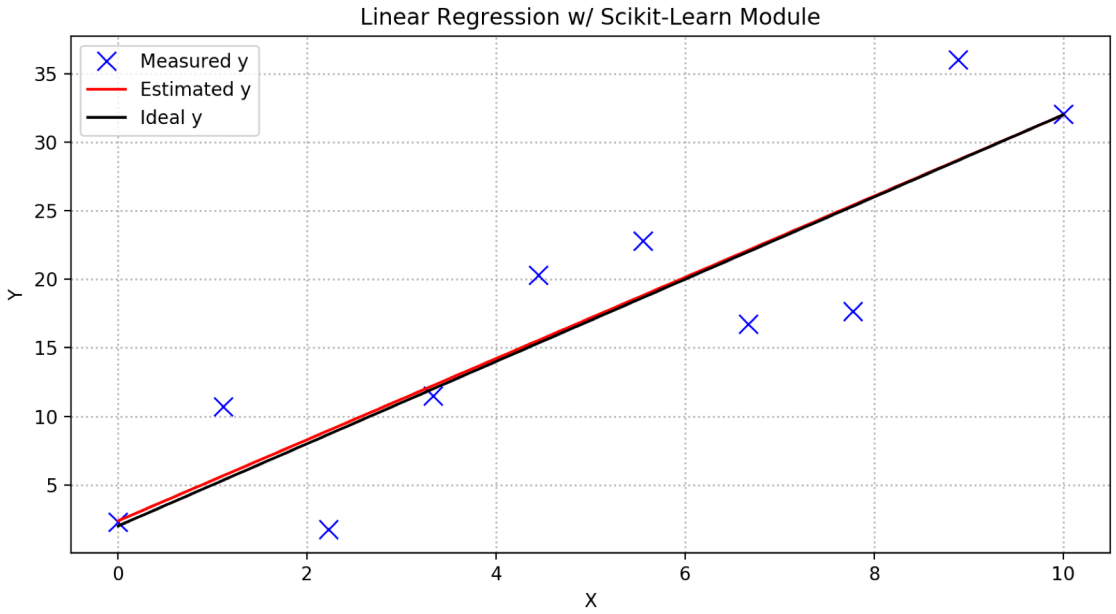


Univariate Linear Regression

$$\hat{y} = \theta_0 + \theta_1 x_1$$

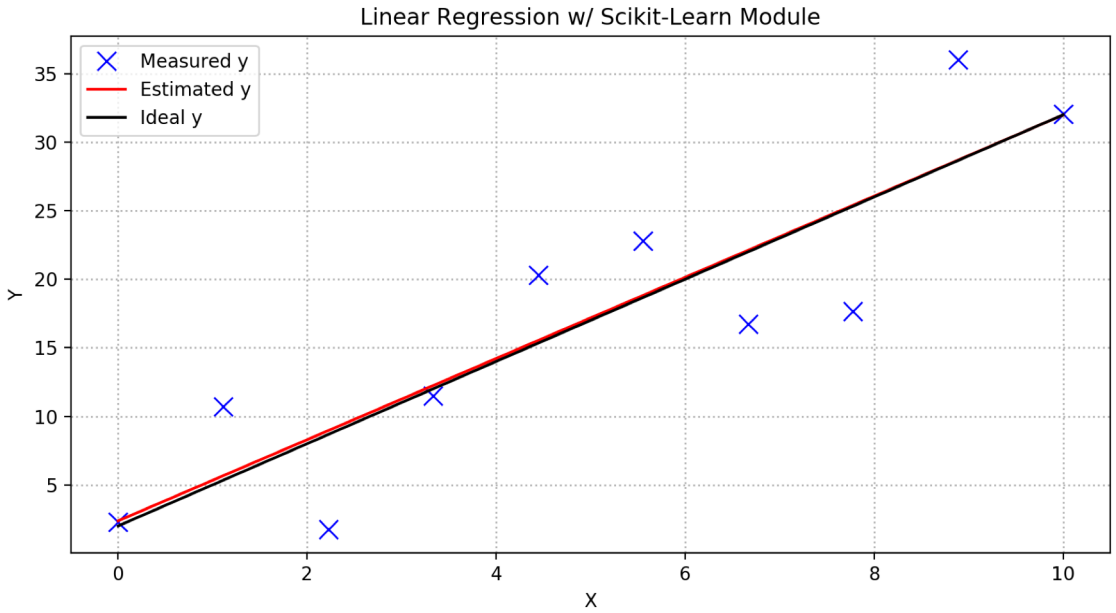
$$\hat{y} = [1 \;\; x_1] \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \Phi\theta$$

(nx1) (nx2) (2x1)



[실습] Scikit-Learn 모듈을 활용한 Regression

```python
# Regression #3 - Scikit-Learn Module

from sklearn import linear_model
import numpy as np
import matplotlib.pyplot as plt

nData = 10
xSampled = np.linspace(0, 10, nData).reshape(-1,1)

yNoisy = np.array([[  2.28093446], [ 10.69146323], [  1.74930661],
                   [ 11.48563487], [ 20.30311448], [ 22.80010189],
                   [ 16.75367794], [ 17.6522528 ], [ 36.0261762 ],
                   [ 32.04787191]])

fitFunc = linear_model.LinearRegression()
fitFunc.fit(xSampled, yNoisy)

xEval = np.linspace(min(xSampled), max(xSampled), 50).reshape(-1,1)
yHat = fitFunc.predict(xEval)

print(fitFunc.coef_)
print(fitFunc.intercept_)
```



[실습] 포유류 Body Weight vs. Brain Weight

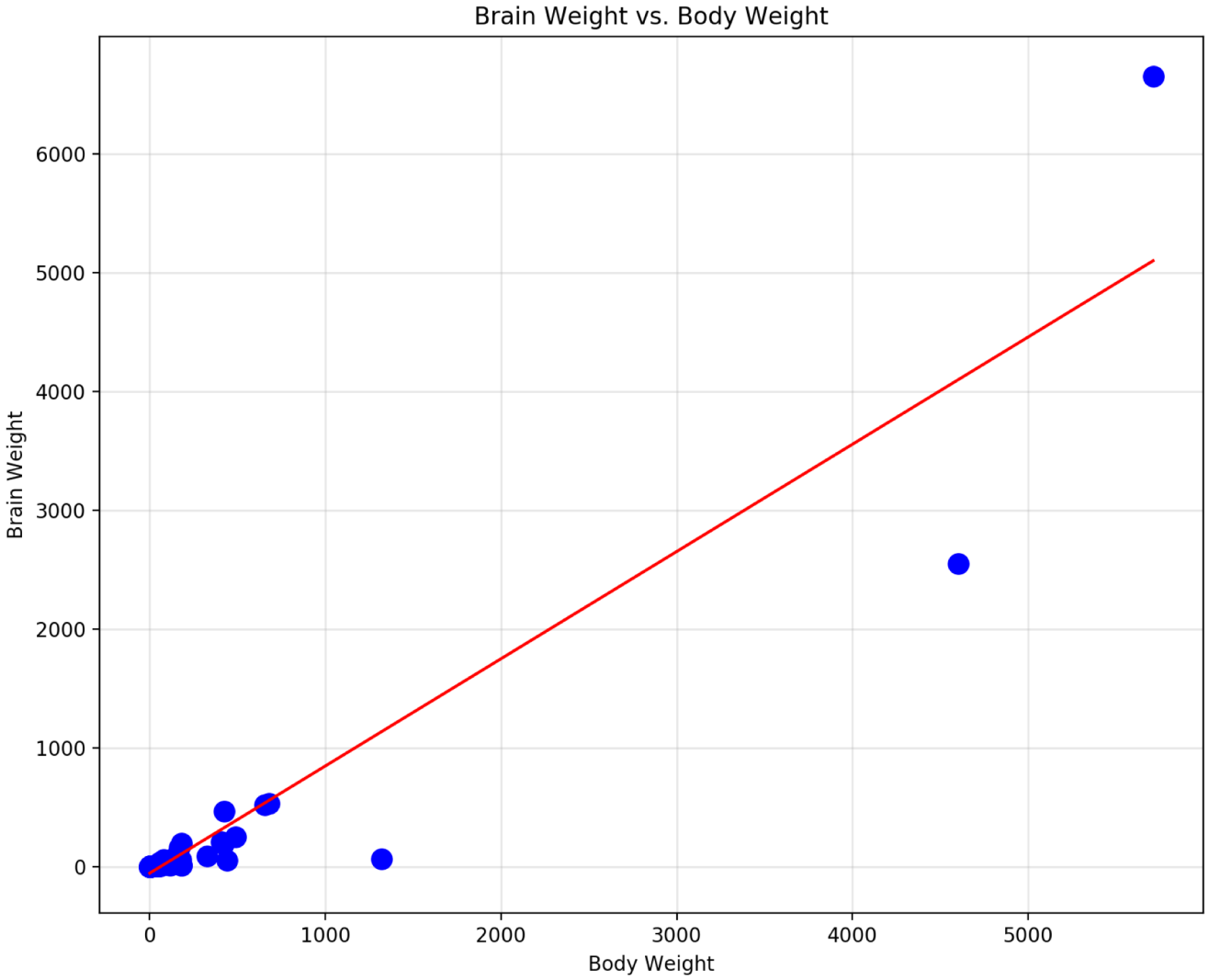
- 다양한 포유동물의 뇌 무게와 몸무게에 관한 dataset이 있다. 두 data 사이의 관계에 대한 선형회귀분석을 진행하시오.
  - Dataset: BrainWeight_BodyWeight.csv
  - Brain_Weight = a*Body_Weight + b

```
# Regression #4 – brain weight vs body weight
from sklearn import linear_model
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dfData = pd.read_csv('BrainWeight_vs_BodyWeight.csv')
dfData.column = ['Brain_Weight', 'Body_Weight']

rawData = dfData.values     # Pandas dataframe을 array로 변환
brainWeight =
bodyWeight =
```

Index	Brain Weight	Body Weight
1	3.385	44.5
2	0.48	15.5
3	1.35	8.1
4	465	423
5	36.33	119.5
6	27.66	115
7	14.83	98.2
8	1.04	5.5
9	4.19	58
10	0.425	6.4
11	0.101	4
12	0.92	5.7
13	1	6.6
14	0.005	0.14
15	0.06	1
16	3.5	10.8
17	2	12.3
18	1.7	6.3
19	2547	4603
20	0.023	0.3
21	187.1	419
22	521	655
23	0.785	3.5
24	10	115
25	3.3	25.6

Dataset source:
https://people.sc.fsu.edu/~jburkardt/datasets/regression/regression.html

[실습] 포유류 Body Weight vs. Brain Weight

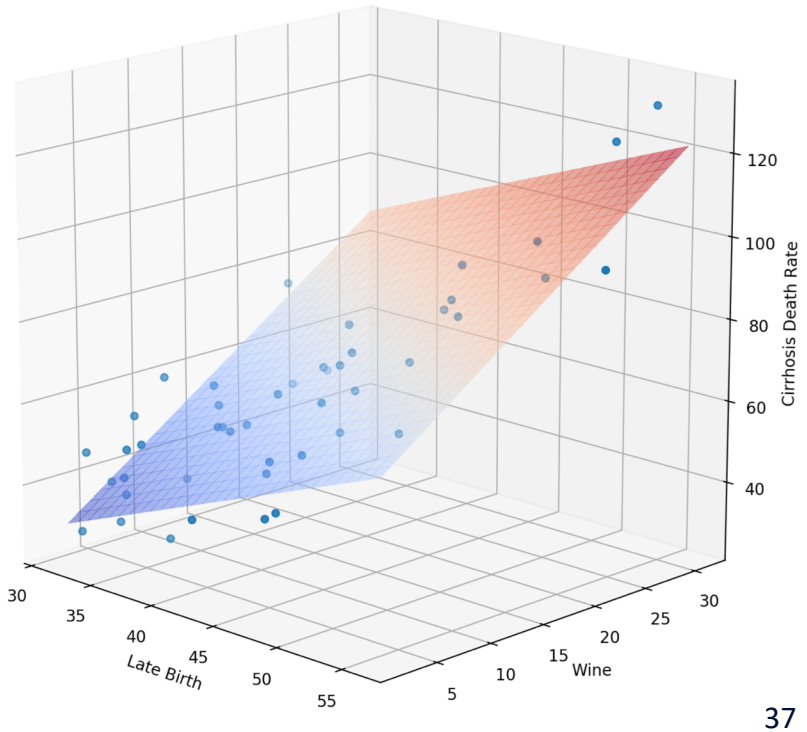


Multivariate Linear Regression

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m$$

$$\hat{y} = [1 \;\; x_1 \; x_2 \cdots x_m] \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix} = \Phi\theta$$

(nx1) (nxm) (mx1)



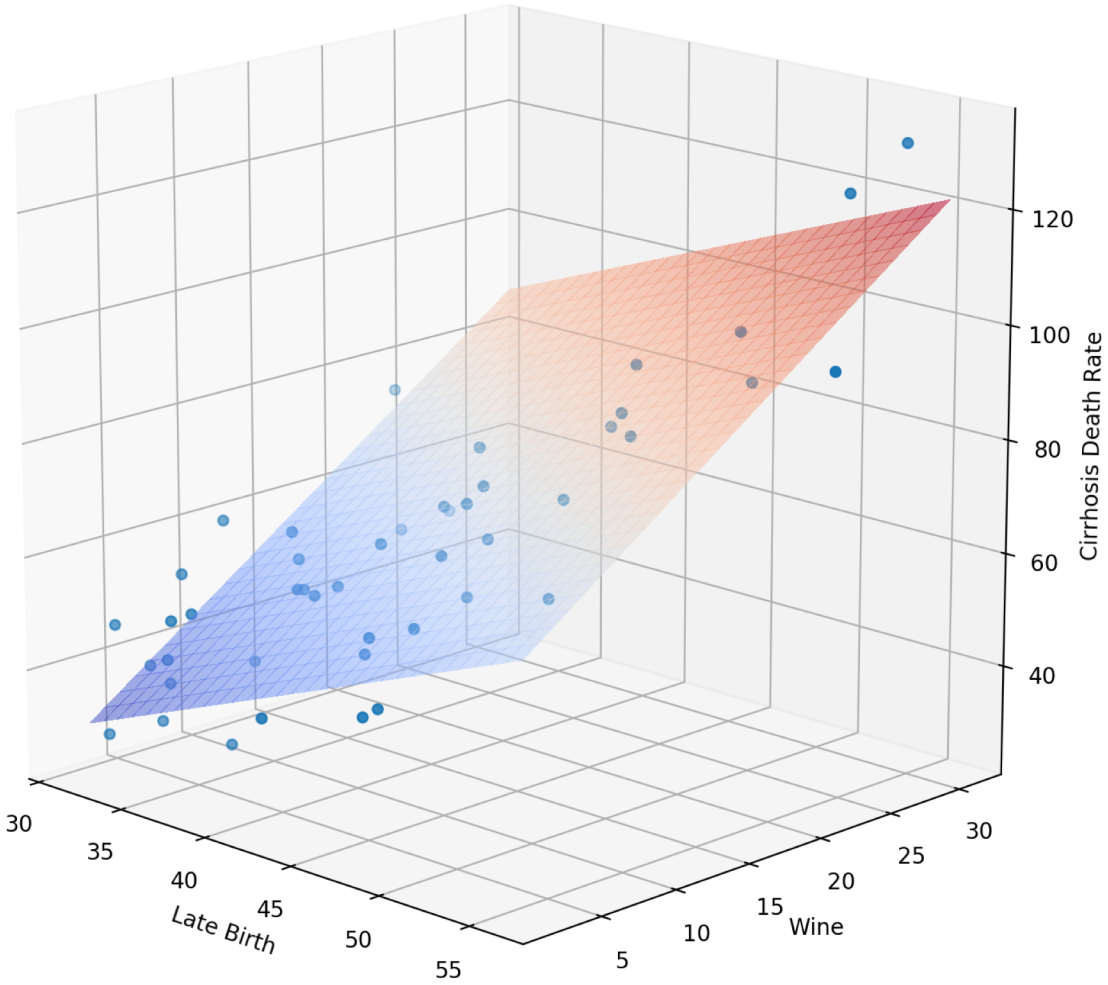
[실습] Multivariate Linear Regression

- 간경변(cirrhosis)에 의한 사망율과 관련된 여러 독립변인들 사이의 관계를 규명하시오.
  - Dataset: population_drinking.csv
  - cirrhosis_death_rate = θ_0 + θ_1*urban_population
    + θ_2*Late_births
    + θ_3*Wine_consumption
    + θ_4*Liquor_comsumption
  - 어떤 독립변인이 cirrhosis_death_rate에 가장 큰 영향을 미치는가?

Index	Urban population (percentage)	Late births (reciprocal * 100)	Wine consumption per capita	Liquor consumption per capita	Cirrhosis death rate
1	44	33.2	5	30	41.2
2	43	33.8	4	41	31.7
3	48	40.6	3	38	39.4
4	52	39.2	7	48	57.5
5	71	45.5	11	53	74.8
6	44	37.5	9	65	59.8
7	57	44.2	6	73	54.3
8	34	31.9	3	32	47.9
9	70	45.6	12	56	77.2
10	54	45.9	7	57	56.6

Dataset source:
https://people.sc.fsu.edu/~jburkardt/datasets/regression/regression.html

[실습] Multivariate Linear Regression

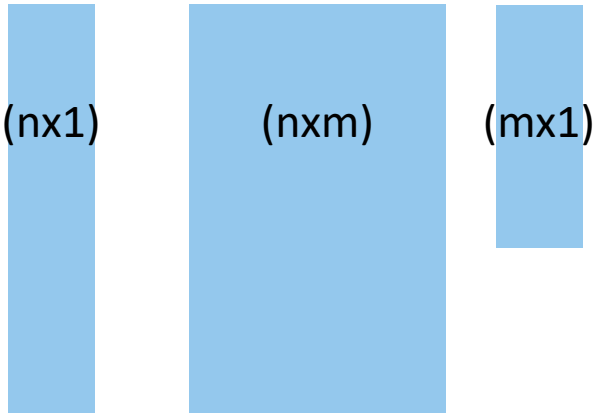


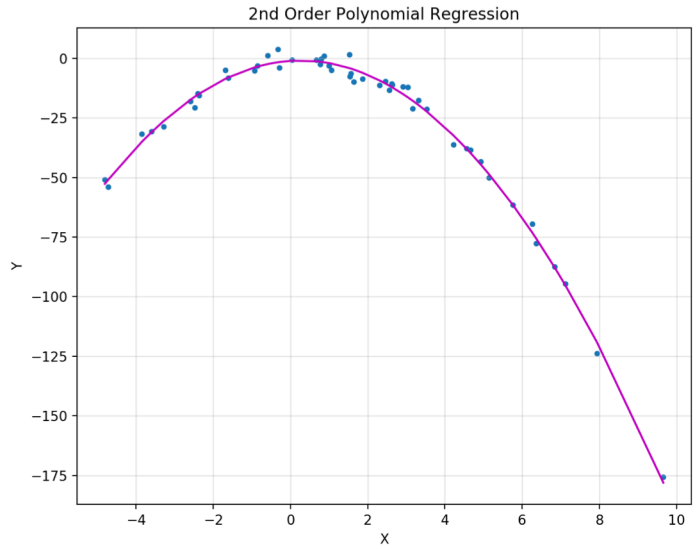
3D Surface Plot
https://matplotlib.org/3.1.3/gallery/mplot3d/surface3d.html

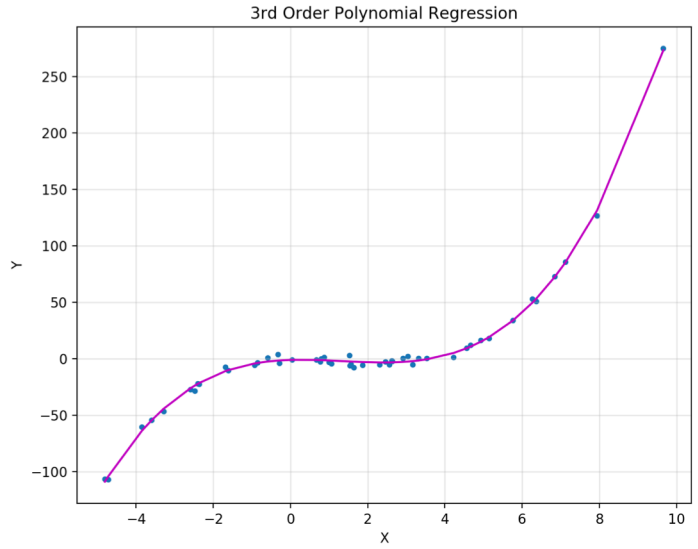
Nonlinear Regression - Polynomial

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

$$\hat{y} = [1 \;\; x_1 \;\; x_1^2] \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \Phi\theta$$





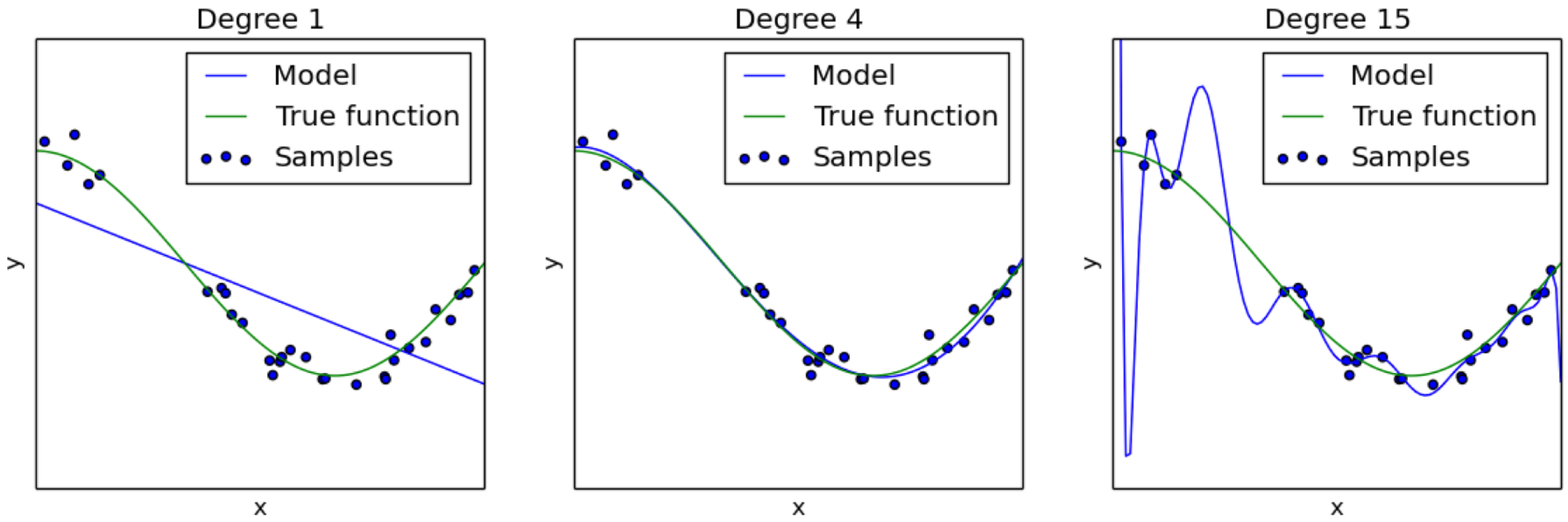


https://towardsdatascience.com/polynomial-regression-bbe8b9d97491

[실습] Polynomial Regression

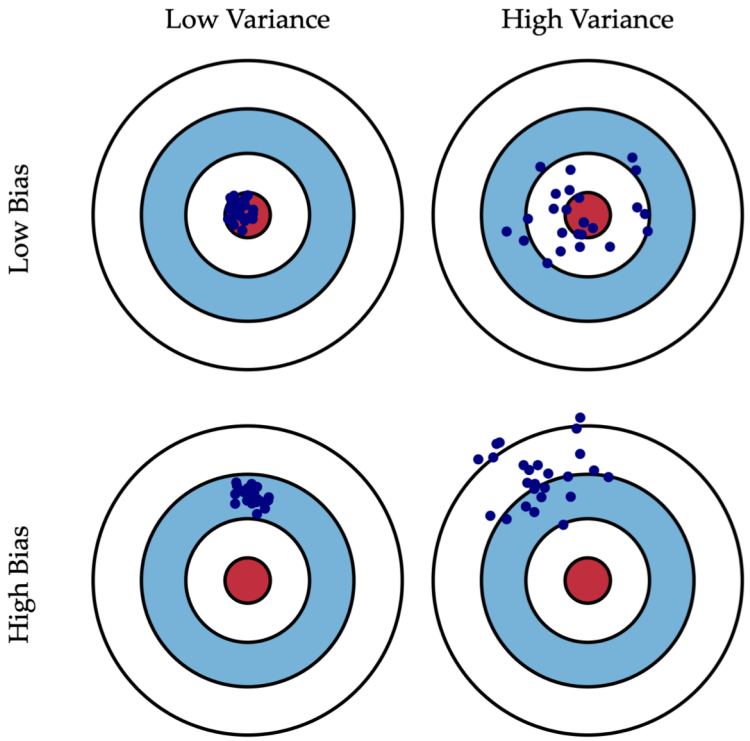
```python
# Polynomial Regression
# Source: https://towardsdatascience.com/polynomial-regression-bbe8b9d97491
import operator

import numpy as np
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures

np.random.seed(0)
nData = 50
x = 2 - 3 * np.random.normal(0, 1, nData)
y = 2 + x - 2 * (x ** 2) + 0.5 * (x ** 3) + np.random.normal(-3, 3, nData)
#y = 2 + x - 2 * (x ** 2) + np.random.normal(-3, 3, nData)

# transforming the data to include another axis
x = x[:, np.newaxis]
y = y[:, np.newaxis]

polynomial_features= PolynomialFeatures(degree=3)
x_poly = polynomial_features.fit_transform(x)

model = LinearRegression()
model.fit(x_poly, y)
y_poly_pred = model.predict(x_poly)

plt.cla()
plt.scatter(x, y, s=10)
# sort the values of x before line plot
sort_axis = operator.itemgetter(0)
sorted_zip = sorted(zip(x,y_poly_pred), key=sort_axis)
x, y_poly_pred = zip(*sorted_zip)
plt.plot(x, y_poly_pred, color='m')
plt.grid(True, alpha = 0.3, color='0.7', linestyle='-', linewidth=1)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('3rd Order Polynomial Regression')
plt.show()
```

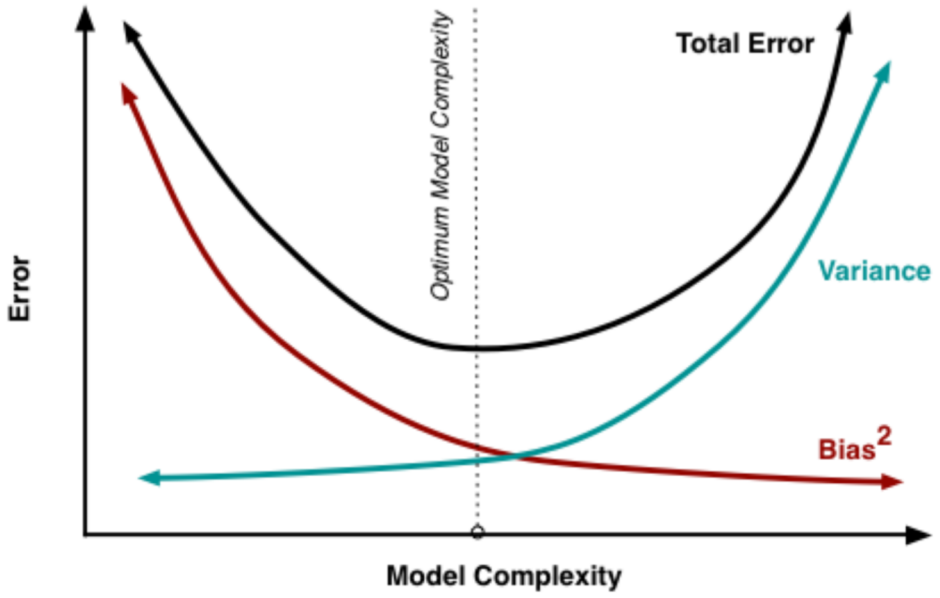
Overfitting & Underfitting



https://scikit-learn.org/0.15/auto_examples/plot_underfitting_overfitting.html

Overfitting & Underfitting





$$Err(x) = E\left[(Y - \hat{f}(x))^2\right]$$

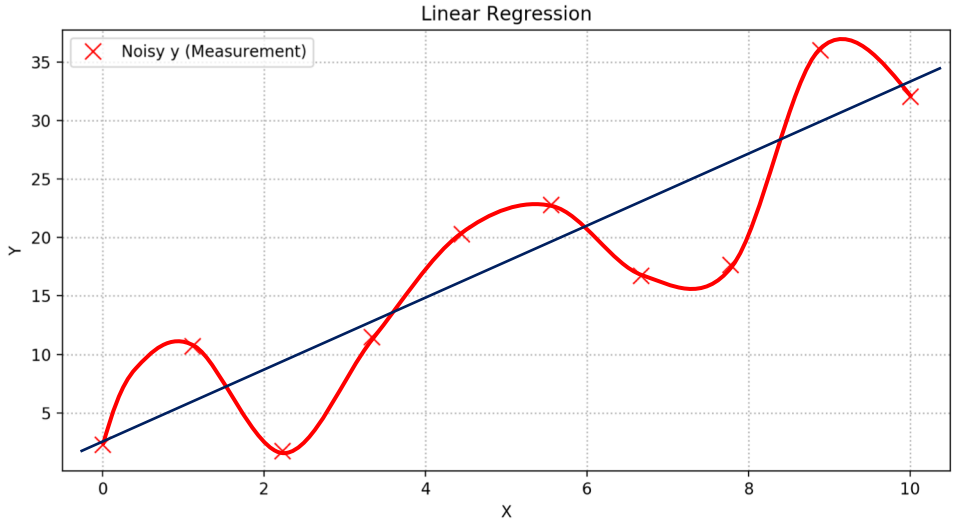
$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

http://scott.fortmann-roe.com/docs/BiasVariance.html

Regularization

- Regularization
  - J(θ) 에 shrinkage penalty (i.e., $\lambda\|\theta\|_1$ 혹은 $\lambda\|\theta\|_2$)를 추가하여, 최적화 과정에서 과적합 방지를 유도함
  - $\lambda\|\theta\|_1$: Lasso regression, $\lambda\|\theta\|_2$: Ridge regression



$$J(\theta) = \sum_{i=1}^{N} (\hat{y}(x_i) - y_i)^2 + \lambda\|\theta\|_2^2$$

$$= \varepsilon^T\varepsilon + \lambda\theta^T\theta$$

$$= (y - \Phi\theta)^T(y - \Phi\theta) + \lambda\theta^T\theta$$

J(θ)의 최소값은 아래 조건을 만족할 때이므로

$$\frac{dJ}{d\theta} = 0 \quad \Rightarrow \quad \theta^* = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T y$$

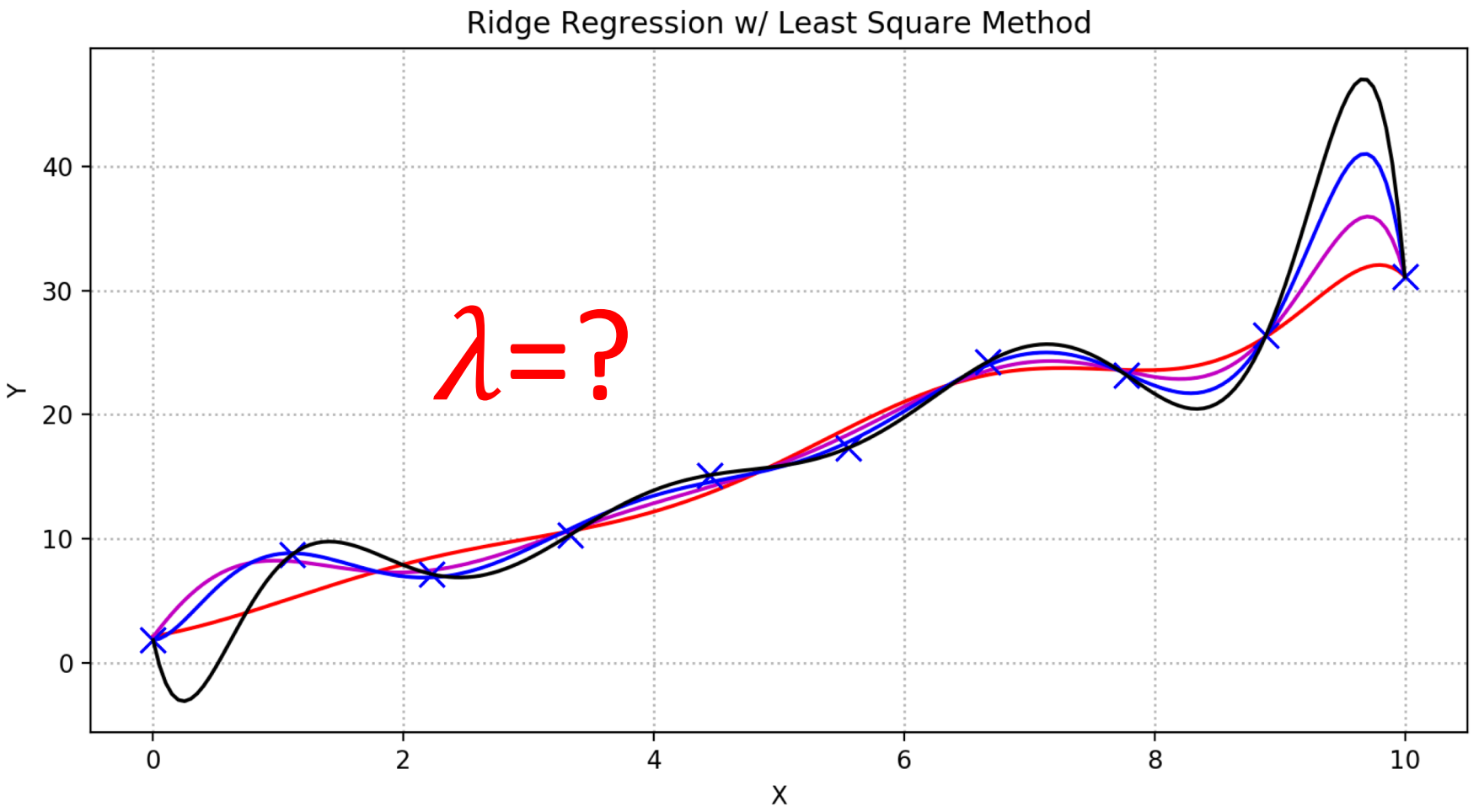
[실습] Ridge Regression – Least Square Method

1. Day2_ML_Regression_03.py 를 수정하여 9차 다항식의 회귀곡선을 구하시오.

2. Lamda를 변화시킬 때, 회귀곡선의 계수에 어떤 변화가 있는지 고려하시오.

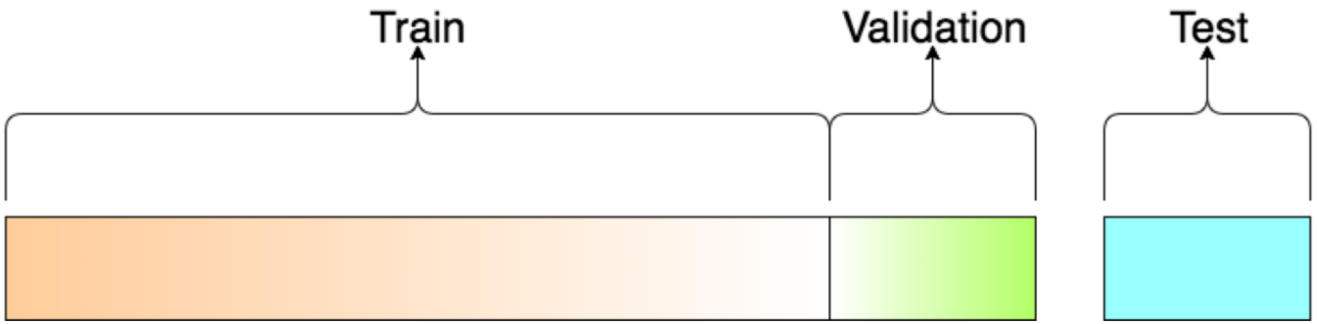
```
57 Phi = np.concatenate([np.ones([nData,1]),
58
59
60                                                    , axis=1)
61
62 Phi = np.asmatrix(Phi)
63
64 lamda_ = 0
65 theta = (                                          ).I  * Phi.T  * yNoisy
```

[실습] Ridge Regression – Least Square Method



Data Split

- Training dataset
  - Model parameter 학습 시 사용
- Validation dataset
  - Hyperparameter tuning 하면서 model fit evaluation 시 활용
- Test dataset
  - Model fit의 최종 평가를 위해 사용
  - 학습 시 사용되지 않음



https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

Time-Series Estimation

ARX model — AutoRegressive w/ eXogenous terms

$$y(t) + a_1 y(t-1) + a_2 y(t-2) + \cdots + a_{n_a} y(t-n_a)$$
$$= b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{n_b} u(t-n_b) + e(t)$$

if $n_a \geq n_b$, $N \gg \max(n_a, n_b)$

$$y(n_a) + a_1 y(n_a-1) + a_2 y(n_a-2) + \cdots + a_{n_a} y(0)$$
$$= b_1 u(n_a-1) + b_2 u(n_a-2) + \cdots + b_{n_b} u(n_a-n_b) + e(n_a)$$

$$y(N) + a_1 y(N-1) + a_2 y(N-2) + \cdots + a_{n_a} y(N-n_a)$$
$$= b_1 u(N-1) + b_2 u(N-2) + \cdots + b_{n_b} u(N-n_b) + e(N)$$

$$\begin{bmatrix} y(n_a) \\ y(n_a+1) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} -y(n_a-1) & -y(n_a-2) & \cdots & -y(0) & u(n_a-1) & \cdots & u(n_a-n_b) \\ & & & \textcircled{5} & & & \\ -y(N-1) & \cdots & & -y(N-n_a) & u(N-1) & \cdots & u(N-n_b) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_{n_a} \\ b_1 \\ \vdots \\ b_{n_b} \end{bmatrix} + \begin{bmatrix} e(n_a) \\ \\ \\ e(N) \end{bmatrix}$$

Time-Series Estimation

$$y = \begin{bmatrix} y(n_b), & y(n_b+1), & \cdots & , & y(N) \end{bmatrix}^T$$

$$\begin{bmatrix} y(n_b) \\ y(n_b+1) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} u(n_b-1) & u(n_b-2) & & u(0) \\ u(n_b) & u(n_b-1) & & u(1) \\ & \vdots & & \\ u(N-1) & u(N-2) & & u(N-n_b) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n_b} \end{bmatrix} + \begin{bmatrix} e(n_b) \\ e(n_b+1) \\ \vdots \\ e(N) \end{bmatrix}$$

$$\left(y(t) = b_1 u(t-1) + b_2 u(t-2) \cdots + b_{n_b} u(t-n_b) + e(t) \right)$$

$$\boxed{y = \Phi\theta + e}$$

$$\boxed{N \gg n_b}$$

$\Rightarrow$ can be solved using the ordinary L.S method.

Time-Series Estimation

6.1.2 Equation Error Identification

ex) ARX model.

A FIR model structure with unknown coefficients $b_1, b_2, \cdots, b_{n_b}$ can be written as a linear regression.

$$y(t) = b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{n_b} u(t-n_b) + e$$

$$\hat{y}(t,\theta) = b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{n_b} u(t-n_b)$$

$$= \varphi(t)^T \cdot \theta \qquad \theta = [b_1, b_2, \cdots, b_{n_b}]^T$$

$$\varphi(t) = [u(t-1), u(t-2), \cdots, u(t-n_b)]^T$$

input : $u(0), u(1), \cdots, u(N)$

output : $y(0), y(1), \cdots, y(N)$

$$y(0) = e(0)$$

$$y(1) = b_1 u(0) + e(1)$$

$$y(2) = b_1 u(1) + b_2 u(2) + e(2)$$

$$y(n_b) = b_1 u(n_b-1) + b_2 u(n_b-2) + \cdots + b_{n_b} u(0) + e(n_b)$$

$$y(N) = b_1 u(N-1) + b_2 u(N-2) + \cdots + b_{n_b} u(N-n_b) + e(N)$$

Time-Series Estimation

ARX model

· AutoRegressive w/ eXogenous terms

$$y(t) + a_1 y(t-1) + a_2 y(t-2) + \cdots + a_{n_a} y(t-n_a)$$
$$= b_1 u(t-1) + b_2 u(t-2) + \cdots + b_{n_b} u(t-n_b)$$
$$+ e(t)$$

ex) $m\ddot{y} + c\dot{y} + ky = f(t)$, $\ddot{u}(t)$, $\dot{u}(t)$

$$A(q) = \sum_{k=0}^{n_a} a_k \cdot q^{-k} \quad (a_0 = 1)$$

$$B(q) = \sum_{k=1}^{n_b} b_k \cdot q^{-k}$$

$$\Rightarrow A(q) \cdot y(t) = B(q) \cdot u(t) + e(t)$$

$$y(t) = \frac{B(q)}{A(q)} u(t) + \frac{1}{A(q)} e(t)$$

$$\Rightarrow ARX(n_a, n_b, n_k)$$

Time-Series Estimation

$y(0)$	$y(1)$	$y(2)$	$y(3)$	$\cdots$	$y(N)$
$u(0)$	$u(1)$	$u(2)$			$u(N)$

sampled data

ARX(3, 3, 1) ← structure

$$y(t) + a_1 y(t-1) + a_2 y(t-2) + a_3 y(t-3) = b_1 u(t-1) + b_2 u(t-2) + b_3 u(t-3) + e(t)$$

$$y(0) = e(0)$$

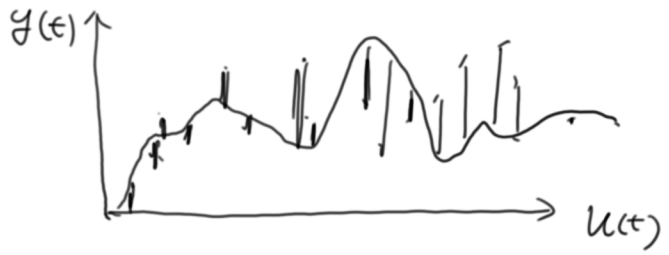
$$y(1) + a_1 y(0) = b_1 u(0) + e(1)$$

$$y(2) + a_1 y(1) + a_2 y(0) = b_1 u(1) + b_2 u(0) + e(2)$$

$$y(3) + a_1 y(2) + a_2 y(1) + a_3 y(0) = b_1 u(2) + b_2 u(1) + b_3 u(0) + e(3)$$

$$\vdots$$

$$y(N) + a_1 y(N-1) + a_2 y(N-2) + a_3 y(N-3) = b_1 u(N-1) + b_2 u(N-2) + b_3 u(N-3) + e(N)$$



Time-Series Estimation Model

- https://machinelearningmastery.com/autoregression-models-time-series-forecasting-python/

- https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/