

ML 툴 비교

부재: Linear Model

김태완

@taewanme

<https://taewan.kim>

ORACLE®

2018.06 SE All Seminar – AI/ML on OCI

김태완과 비슷한 SE를 찾아라: 유사도 측정 결과

	김태완	강인호	이동욱	최수현	나지훈
김태완	0.589	0.707	0.133	0.133	
K-Nearest Neighbor Algorithm		K-NN			
SC-002	0.707	0.44	0.251	0.251	
SC-003	0.133	0.251	0.251		0.714
SC-072	0.133	0.251	0.251	0.714	

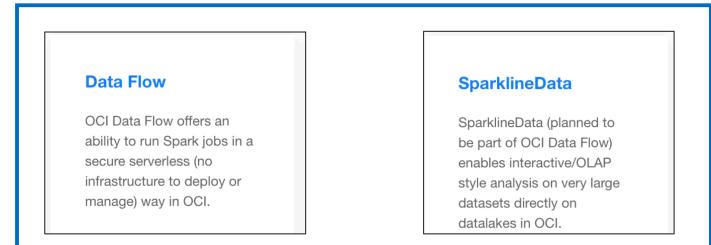
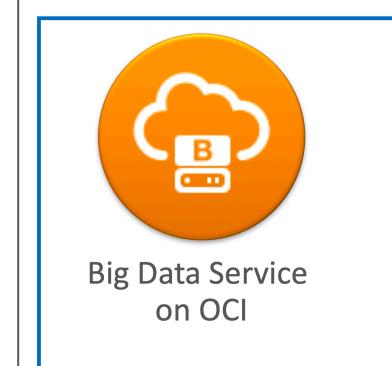
ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. | Confidential – Oracle Internal/Restricted/Highly Restricted

23

2018.07 SE All Seminar – Spark on OCI

Oracle Big Data Services on OCI



클라우드에서 Big Data를 하고자 하는 고객
Cloud Native Big Data

클라우드에서 Hadoop을 하고 싶은 고객
- On-premise 연동, CDH 활용

ORACLE®

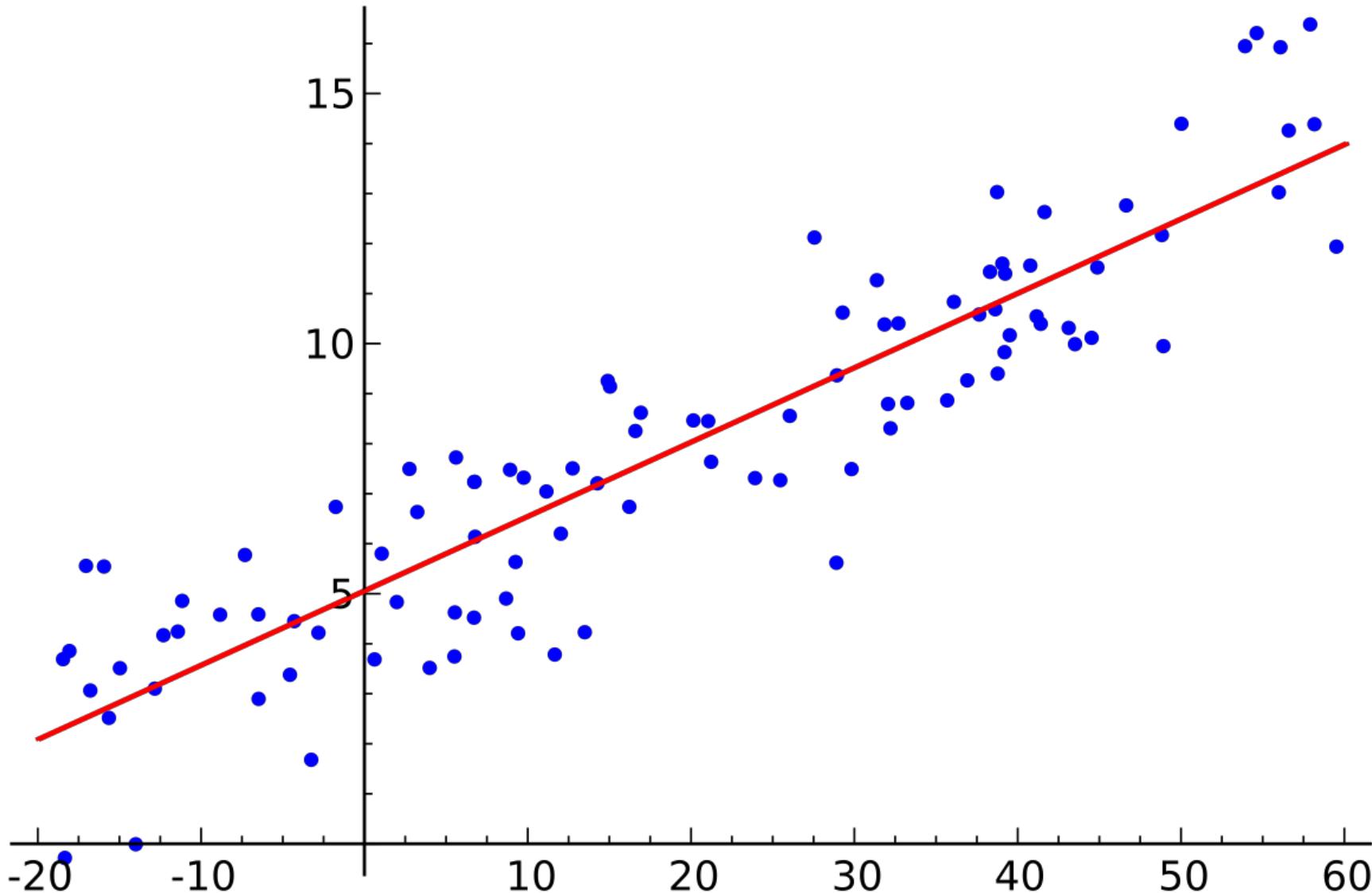
Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

74

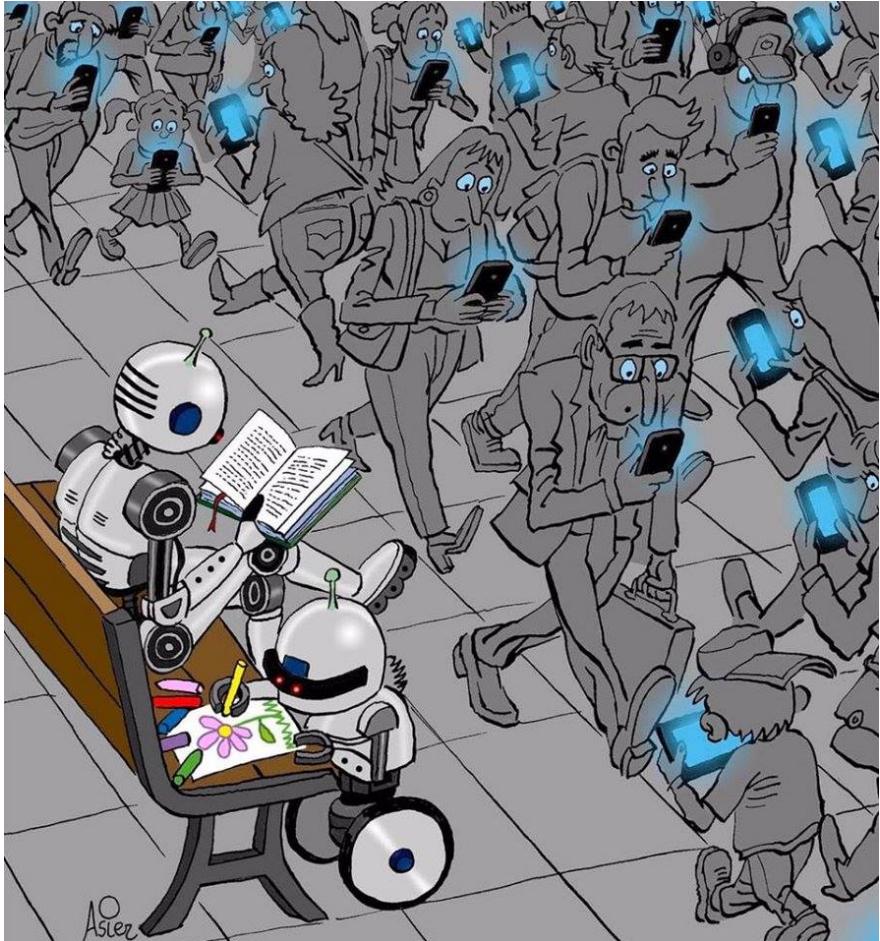
ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. | Confidential – Oracle Internal/Restricted/Highly Restricted

2

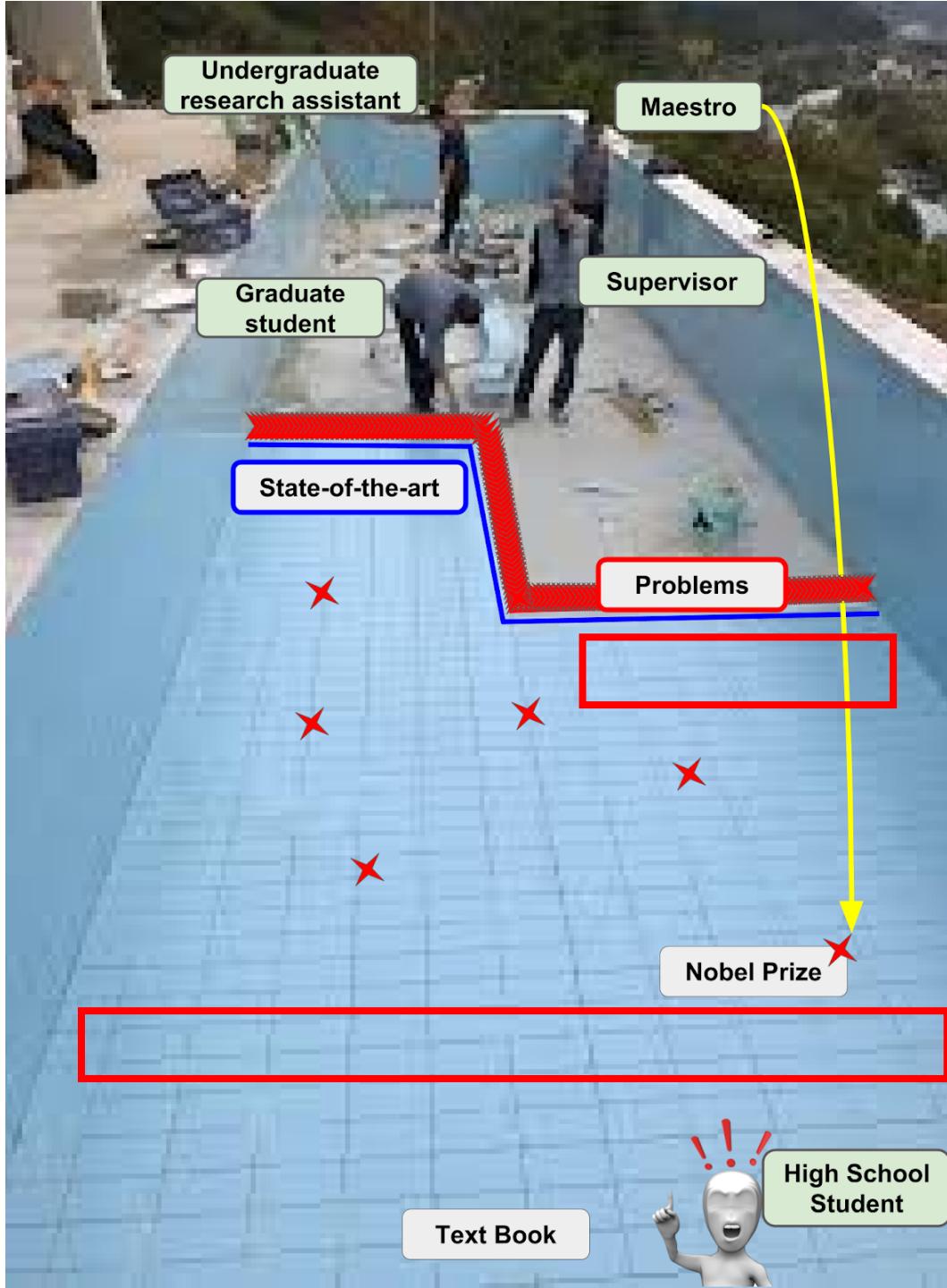


인간과 AI 그리고



- Oracle Position?
- SE Position?

<https://jarche.com/2019/05/a-decade-of-finds/>



Kaggle: <http://kaggle.com>

kaggle Search Competitions Datasets Notebooks Discussion Courses ...

Competitions

General InClass Sort by Grouped All Categories Search competitions

19 Active Competitions

Rank	Competition Name	Description	Prize	Teams
1	Severstal: Steel Defect Detection	Can you detect and classify defects in steel?	\$120,000	672 teams
2	APOTS 2019 Blindness Detection	Detect diabetic retinopathy to stop blindness before it's too late	\$50,000	2,585 teams
3	SIIM-ACR Pneumothorax Segmentation	Identify Pneumothorax disease in chest x-rays	\$30,000	1,417 teams
4	Predicting Molecular Properties	Can you measure the magnetic interactions between a pair of atoms?	\$30,000	2,591 teams

kaggle Search Competitions Datasets Notebooks Discussion Courses ...

Notebooks

Public Your Work Favorites Sort by Hotness Languages Types Tags Search notebooks

Categories Outputs Languages Types Tags

46 ⚡ Python Chemistry of Best Model 1h ago with multiple data sources

58 ⚡ R Criskiev's distances, more 5h ago in champs-scalar-coupling

58 ⚡ Sanfrancisco Crime Analysis 9h ago in Sanfrancisco Crime Dataset geospatial analysis, data visualization

33 ⚡ Julia Immigration to Canada 2h ago in Immigration to Canada IBM Dataset immigration, geospatial analysis, data visualization

12 ⚡ Python Chemistry of Best Models (-1.801) 9h ago with multiple data sources

9 ⚡ Python Beginners guide to MNIST with fast.ai 6h ago in digit-recognizer GPU 0.82985

22 ⚡ Python Comprehensive Data Preprocessing And Modeling 8h ago in house-prices-advanced-regression-techniques tutorial, beginner, data cleaning, feature ...

Kaggle: Boston House Pricing

The screenshot shows the Kaggle website interface. At the top, there is a navigation bar with links for 'Search', 'Competitions', 'Datasets', 'Notebooks', 'Discussion', 'Courses', and more. A user icon and a notification bell are also present. The main content area displays a dataset titled 'Boston House Prices' by 'Manimala'. The dataset is described as a 'Regression predictive modeling machine learning problem from end-to-end Python'. It was updated 2 years ago (Version 1). Below the title, there are tabs for 'Data', 'Kernels (28)', 'Discussion (1)', 'Activity', and 'Metadata'. There is a 'Download (13 KB)' button and a 'New Notebook' button. Below these, there are sections for 'Usability 7.1', 'License Other (specified in description)', and 'Tags No tags yet'. The 'Description' section is empty. The 'Context' section contains the text: 'To Explore more on Regression Algorithm'. The 'Content' section provides a detailed description of the dataset: 'Each record in the database describes a Boston suburb or town. The data was drawn from the Boston Standard Metropolitan Statistical Area (SMSA) in 1970. The attributes are defined as follows (taken from the UCI Machine Learning Repository1): CRIM: per capita crime rate by town 2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft. 3. INDUS: proportion of non-retail business acres per town 4. CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) 5. NOX: nitric oxides concentration (parts per 10⁻⁵) 6. RM: average number of rooms 7. AGE: age of the house 8. DIS: weighted distances to five Boston employment centers 9. RAD: index of accessibility to radial highways 10. TAX: full-value property tax rate per \$10,000 11. PTRATIO: pupil-teacher ratio by town 12. B: 1000(Bk - 0.63)^2 where Bk is the proportion of lower status of the population 13. LSTAT: % lower status of the population 14. MEDV: median value of owner-occupied homes'.

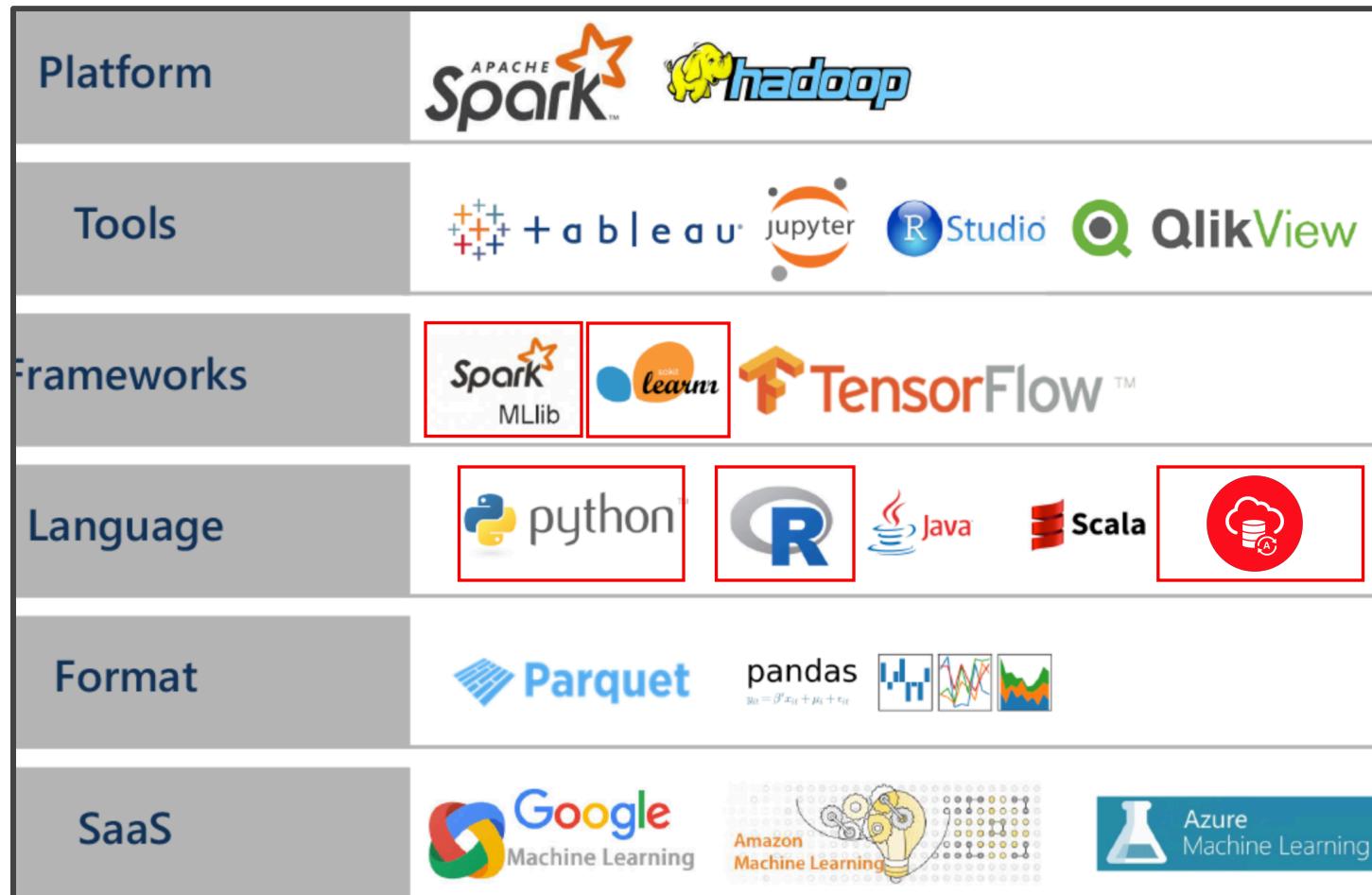
Linear Regression

$$\hat{y} = x_1w_1 + x_2w_2 + \cdots + x_nw_n + b$$

Linear Regression

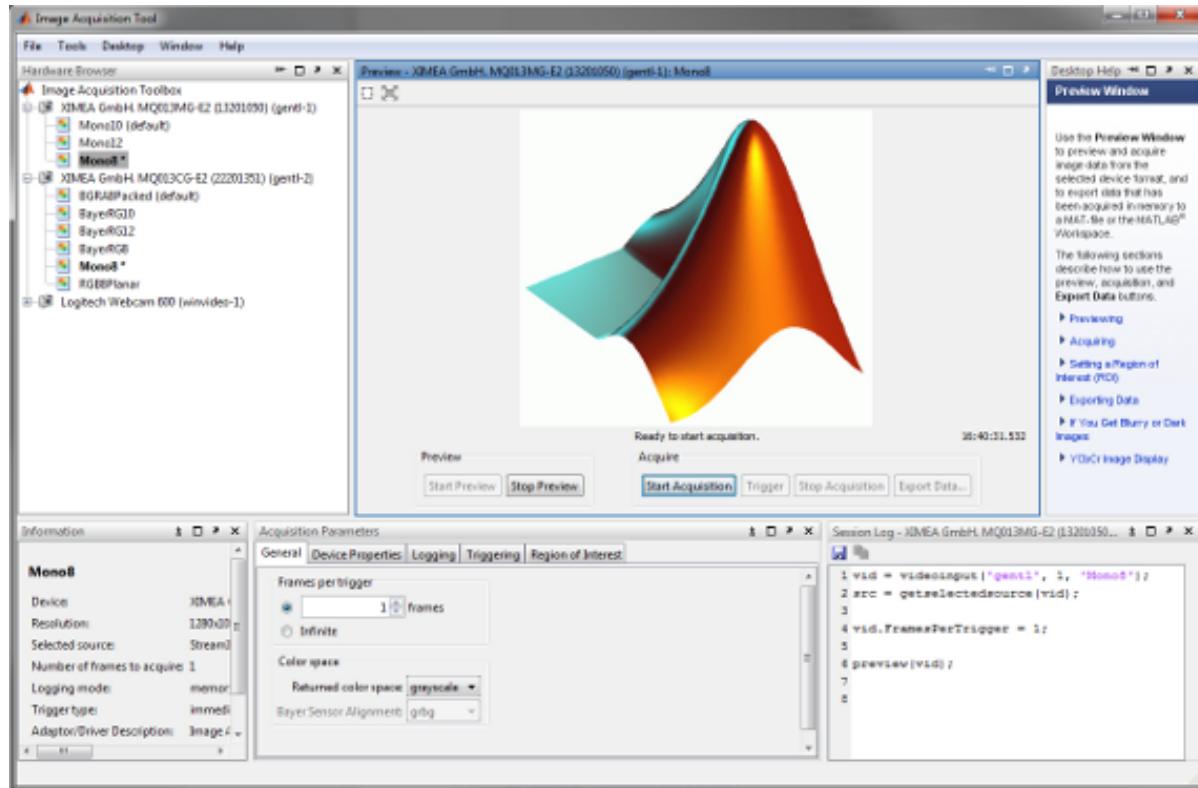
$$\begin{aligned}\hat{y} &= x_1w_1 + x_2w_2 + \cdots + x_nw_n + b \\ &= \sum x_nw_n = W^T\mathbf{x}\end{aligned}$$

데이터 분석 기술



<https://volansys.com/data-science-services/>

분석 기술: Matlab



Matt Lee 6분 ·

방금 뉴욕의 개발자 임백준님의 글에 서도 언급 되어 있지만
Matlab은 친청인 학계에서도 슬슬 기피 당하고 나는 진짜 꼭 서보고 싶었지만 그럴 기회
조차도 없었음.
(파이썬,R짱짱맨이야, 그리고 줄리아 슬금슬금 성장중)

나는 아직도 수치해석 때 교수님께서 하신 말씀을 잊지 못한다.

"우리는 코딩 실습 Matlab으로 하지 않습니다."

"왜요 교수님?"

"여러분은 연계 프로젝트(여기서 연계는 다른 학계나 기업, 정부와의 연계를 뜻함)용 언
어로 쓰지도 못 할 언어를 단 한 두번의 수업을 위해서 배우고 사용하고 싶나요?"
(한 마디로 좀 심하게 말해서 수학과나 물리학과 빼고는 잘 사용하지도 않는 죽은 고인물
의 언어라는 뜻...)

.....

Matlab 지못미ㅠㅠ

그 외 중에 가격은 엄청 비싸서 출입과 동시에 쓰지도 못함;;;

?

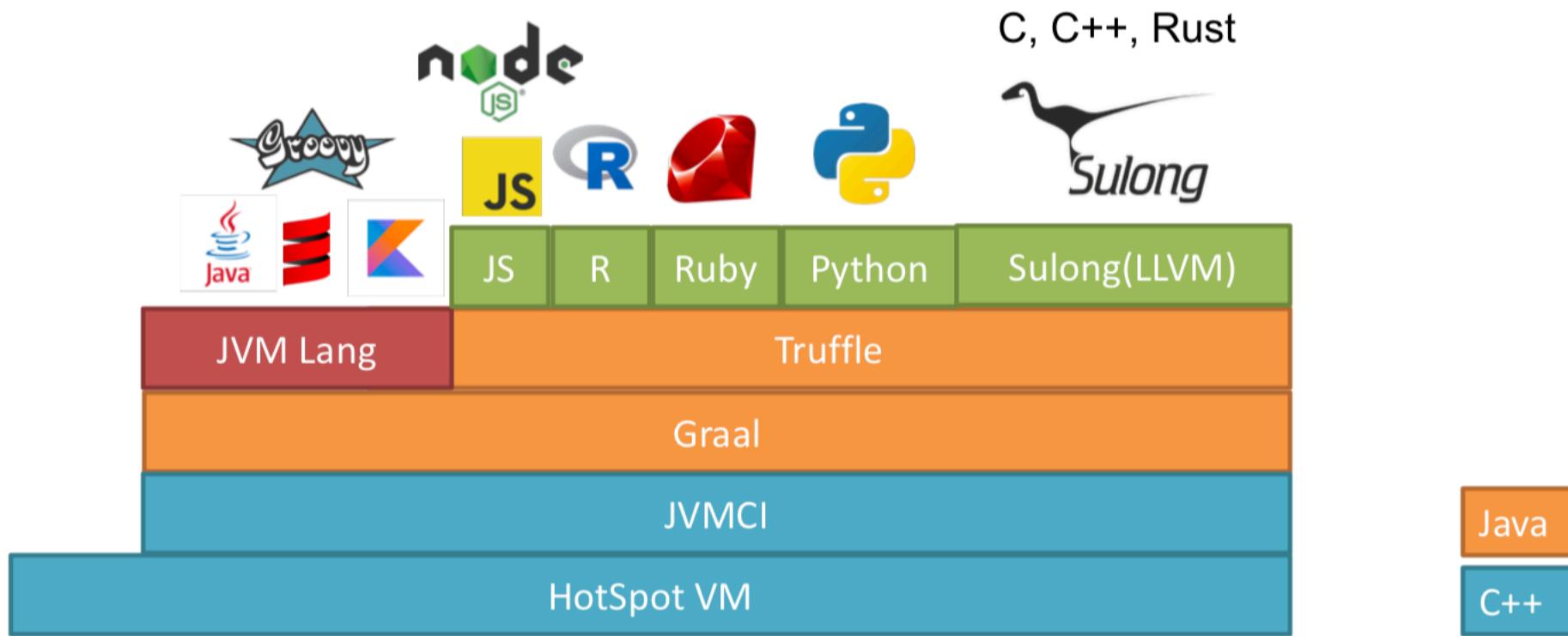
분석 기술:R

데이터 분석 / 시각화를 위한 전용언어



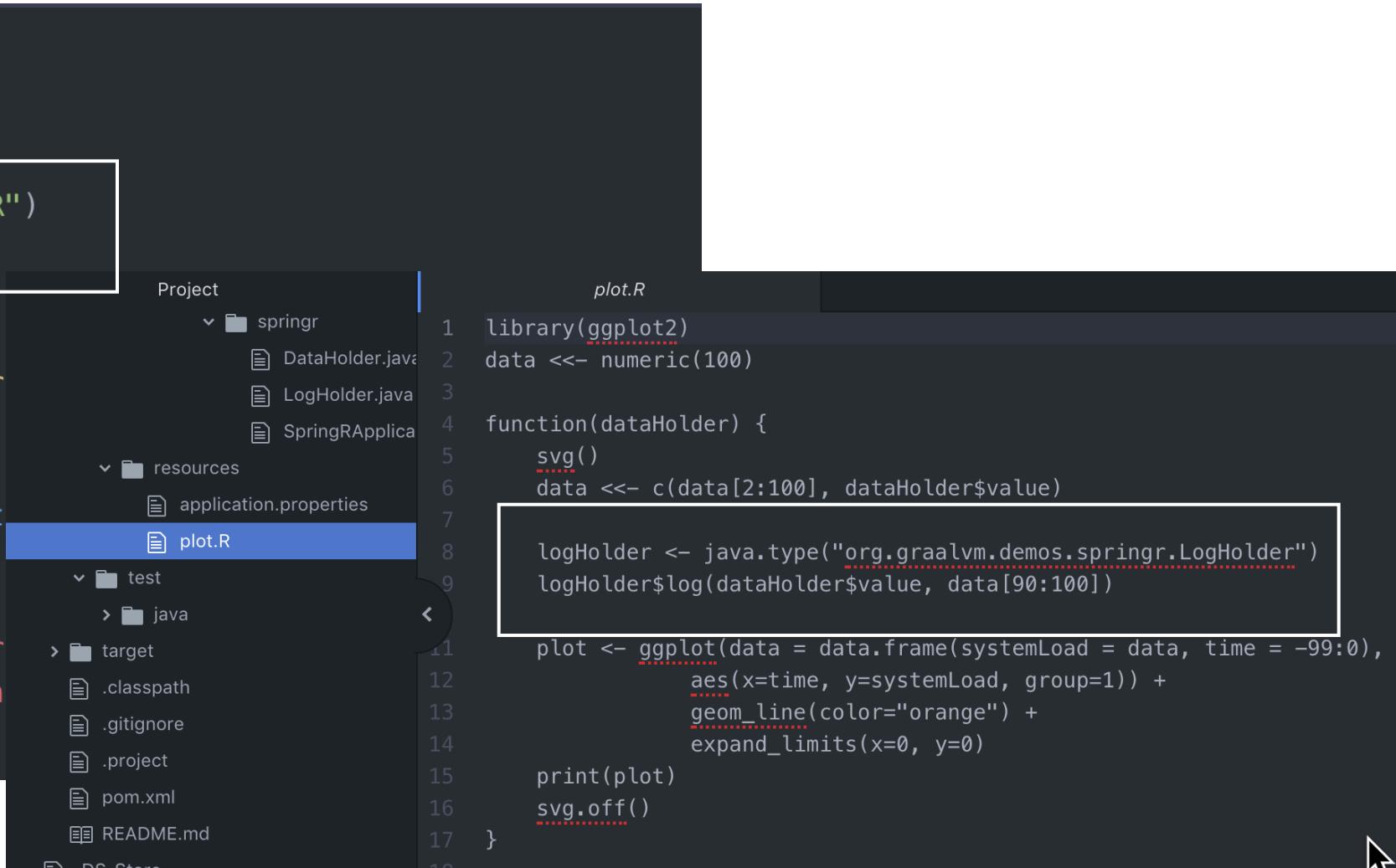
- 학습 용이
- 언어 구조 단순
- 풍부한 알고리즘
- 확장성 결여

GraalVM & Fast R



spring-r: mvn spring-boot:run

```
@Controller  
@SpringBootApplication  
public class SpringRApplication {  
  
    @Value(value = "classpath:plot.R")  
    private Resource rSource;  
  
    @Autowired  
    private Function<DataHolder, String> getPlot;  
  
    @Bean  
    Function<DataHolder, String> getPlot() throws IOException {  
        Source source =  
            Source.newBuilder("R", rSource).build();  
        return ctx.eval(source).as(Function.class);  
    }  
}
```

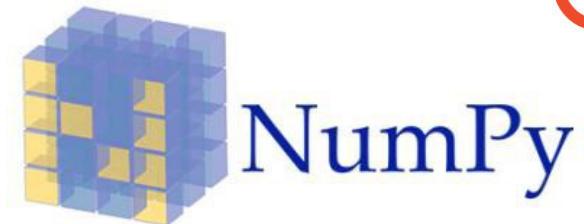


The screenshot shows a Java project structure on the left and an R script on the right. The Java project contains a `spring` package with `DataHolder.java`, `LogHolder.java`, and `SpringRApplication.java`. It also has a `resources` folder containing `application.properties` and `plot.R`. The `plot.R` file is highlighted. The R script defines a function `function(dataHolder) {` that uses `ggplot2` to create a line plot. It includes imports for `ggplot2` and `org.graalvm.demos.springr.LogHolder`, and uses `java.type` to create an instance of `LogHolder`.

```
Project  
  spring  
    DataHolder.java  
    LogHolder.java  
    SpringRApplication.java  
  resources  
    application.properties  
    plot.R  
  test  
  target  
  .classpath  
  .gitignore  
  .project  
  pom.xml  
  README.md
```

```
plot.R  
library(ggplot2)  
data <- numeric(100)  
  
function(dataHolder) {  
  svg()  
  data <- c(data[2:100], dataHolder$value)  
  
  logHolder <- java.type("org.graalvm.demos.springr.LogHolder")  
  logHolder$log(dataHolder$value, data[90:100])  
  
  plot <- ggplot(data = data.frame(systemLoad = data, time = -99:0),  
                 aes(x=time, y=systemLoad, group=1)) +  
        geom_line(color="orange") +  
        expand_limits(x=0, y=0)  
  print(plot)  
  svg.off()  
}
```

분석 기술:Python



TensorFlow

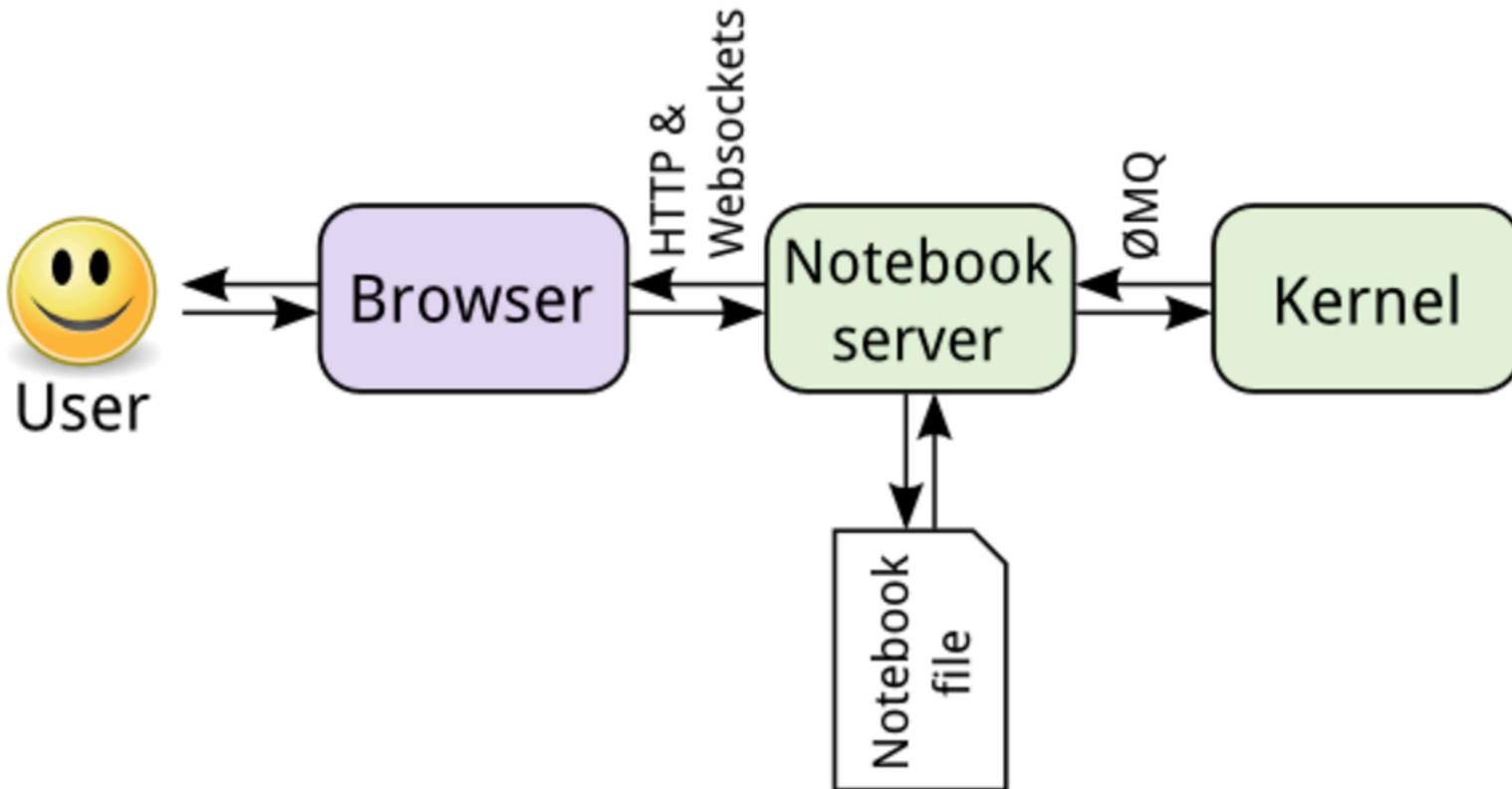


분석 기술: Spark



- Sckit- Learn의 영향
- 복잡성 높음
- 기술 난이도 높음
- 개발 난이도 높음

웹기반 분석환경



웹기반 분석환경

Jupyter spectrogram (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#)

$$X_k = \sum_{n=0}^{N-1} x_n \exp\left(\frac{-2\pi i}{N} kn\right) \quad k = 0, \dots, N-1$$

```
In [2]: from scipy.io import wavfile  
rate, x = wavfile.read('test_mono.wav')
```

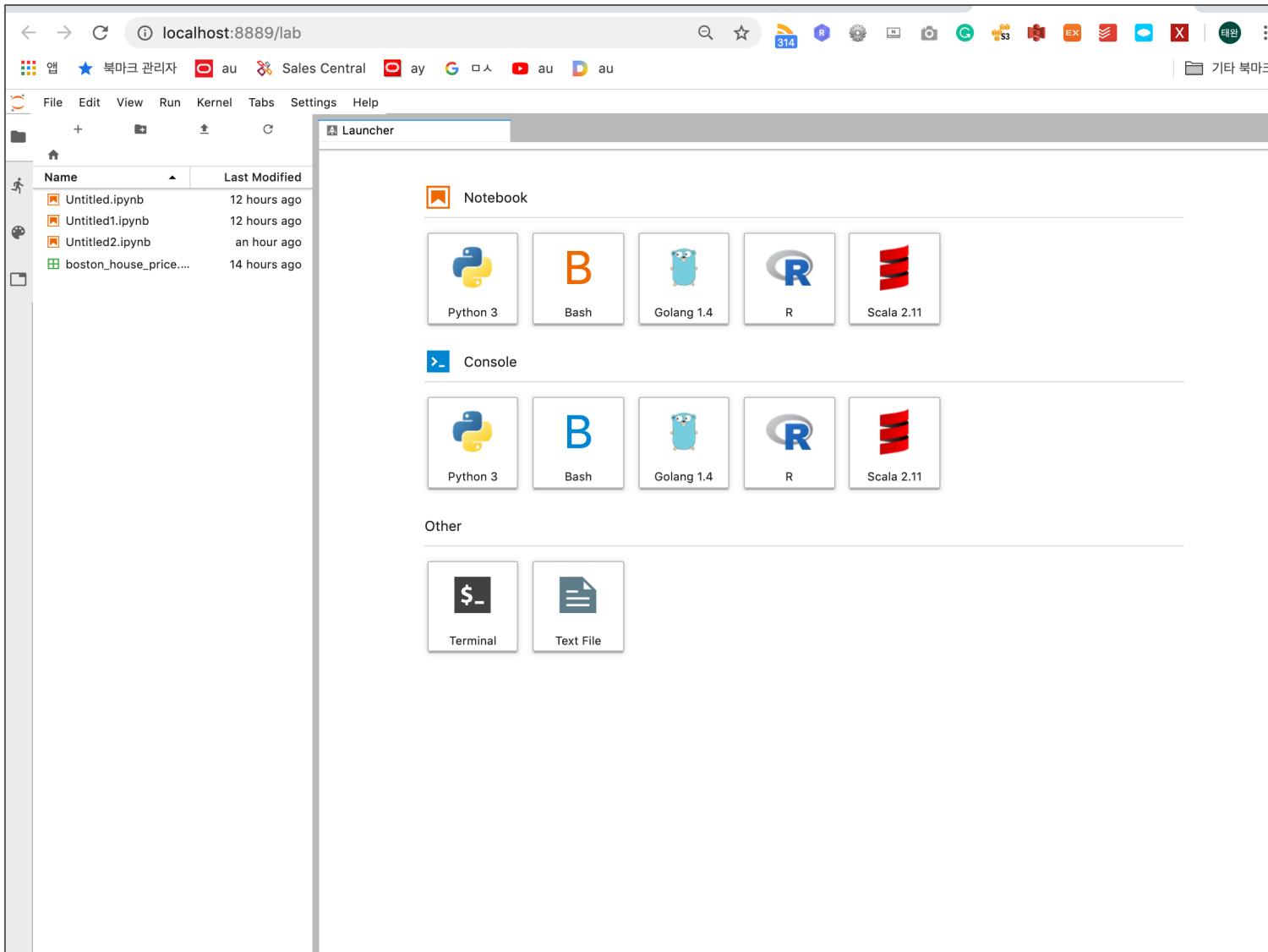
And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

```
In [5]: fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))  
ax1.plot(x); ax1.set_title('Raw audio signal')  
ax2.specgram(x); ax2.set_title('Spectrogram');
```

The figure consists of three side-by-side data visualization panels, each with a search bar and a 'FINISHED' button at the top.

- Left Panel:** A donut chart titled "Under age < 35" showing the distribution of ages. The legend lists ages from 19 to 34. The chart shows a significant concentration of younger individuals, with the largest segments being 34 (34%) and 33 (33%).
- Middle Panel:** A bar chart titled "Under age < 35" showing the count of individuals by age. The x-axis shows ages 22, 26, and 30. The y-axis ranges from 0 to 103. The data is represented as grouped bars, with values increasing from approximately 5 for age 22 to a peak of about 103 for age 30.
- Right Panel:** A line chart titled "marital" showing the value of marital status. The x-axis shows ages 19, 40, 60, and 69. The y-axis ranges from 1 to 105. The data is represented as a single blue line, showing a sharp increase from age 19 to a peak around age 35, followed by a gradual decline.

웹기반 분석환경: Jupyter Lab



웹기반 분석환경: Jupyter >> Zeppelin

A screenshot of a GitHub file page. The URL is <https://github.com/oracle/learning-library/blob/master/workshops/journey4-adwc/files/Basic%20Machine%20Learning.json>. The page shows a single line of JSON code:

```
{"paragraphs": [{"text": "# Basic Machine Learning using DBMS_PREDICTIVE_ANALYTICS\n\nOne of the"}]}
```

The GitHub interface includes a sidebar with a user icon, a search bar, and navigation links like Pull requests, Issues, Marketplace, and Explore. The main content area shows the file's details, contributors, and raw code.

A screenshot of a GitLab notebook titled 'Analysis.ipynb'. The code cell contains the following Python code:

```
%matplotlib inline

import os
from pprint import pprint

# Only needed to embed the images in notebook
from IPython.display import Image

import numpy as np
import matplotlib.pyplot as plt

from srim import TRIM, SR, Ion, Layer, Target
from srim.output import Results
```

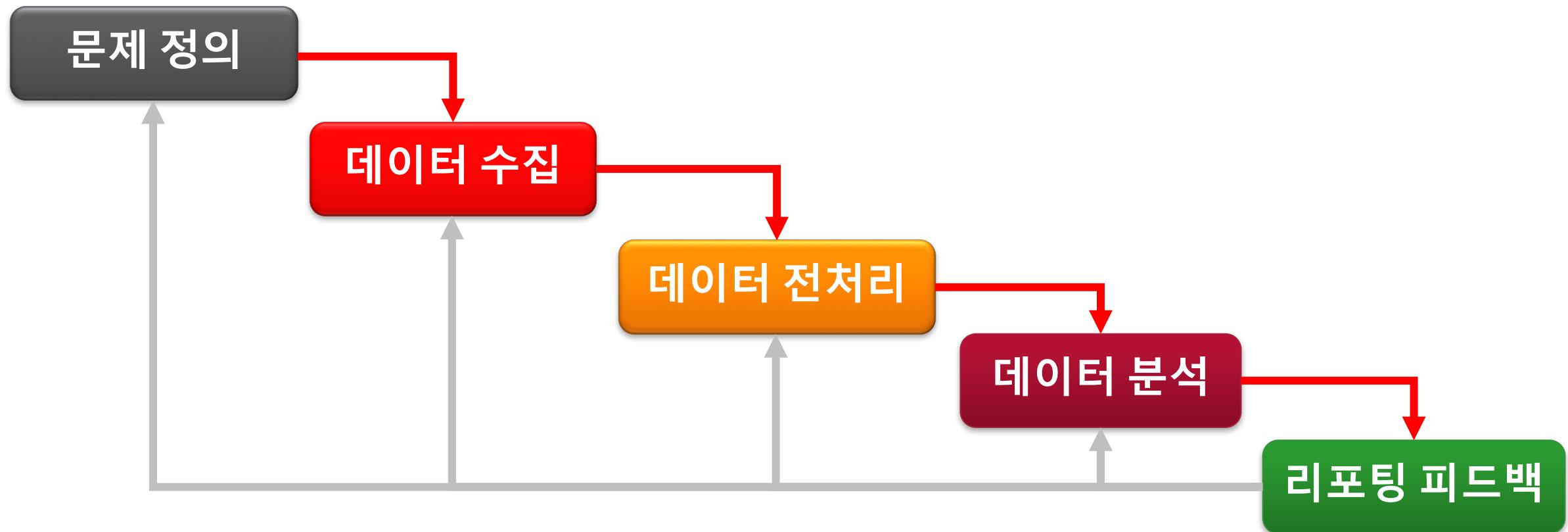
The notebook interface includes a sidebar with a user icon, a search bar, and navigation links like Projects, Groups, Snippets, and Help. Below the code cell, there is a section titled 'Run a SRIM Calculation' with explanatory text and a list of concepts:

- a list of Layers forms a Target
- a Layer is a dict of elements, with density, and a width
- an Element can be specified by symbol, atomic number, or name, with a custom mass [amu]
- an Ion is like an Element except that it also requires an energy in [eV]

<https://github.com/oracle/learning-library/blob/master/workshops/journey4-adwc/files/Basic%20Machine%20Learning.json>

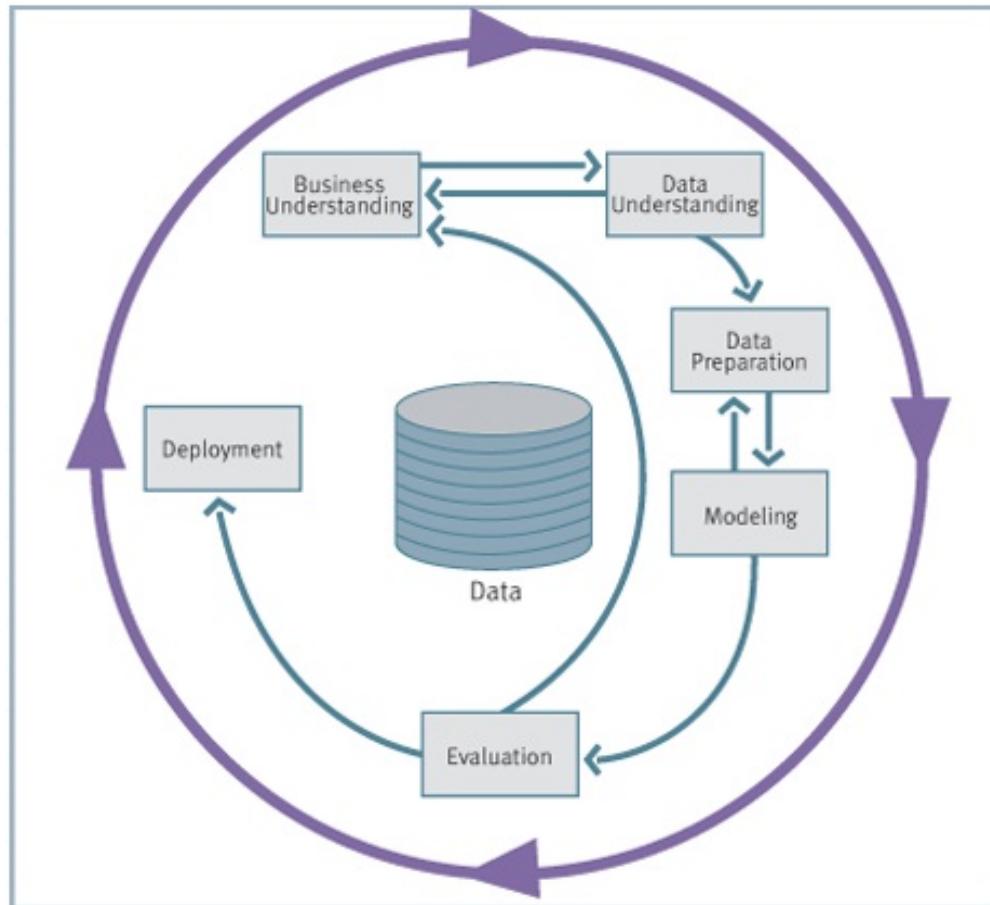
https://gitlab.com/costrouc/pysrim/blob/master/examples/not_ebooks/Analysis.ipynb

데이터 분석 프로세스

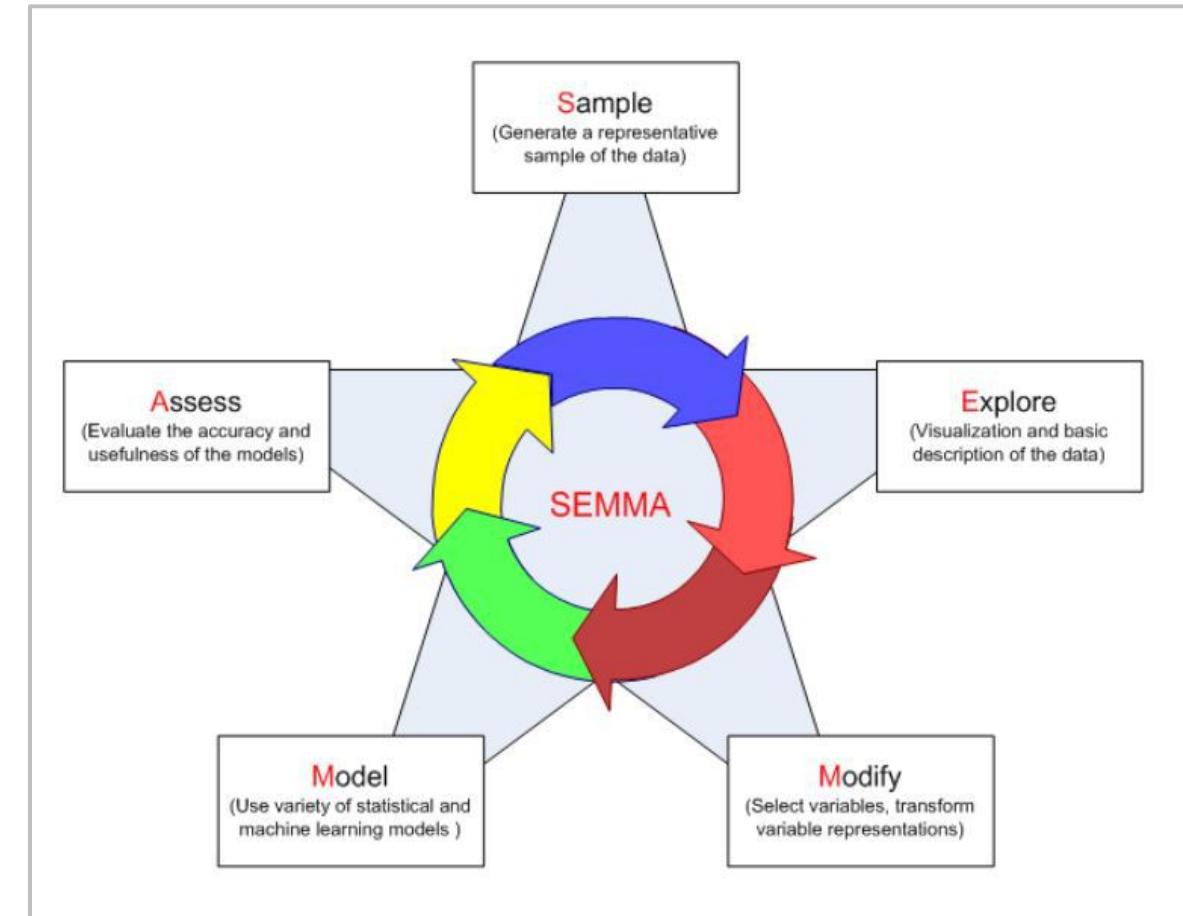


데이터 분석 프로세스: CRISP-DM, SEMMA

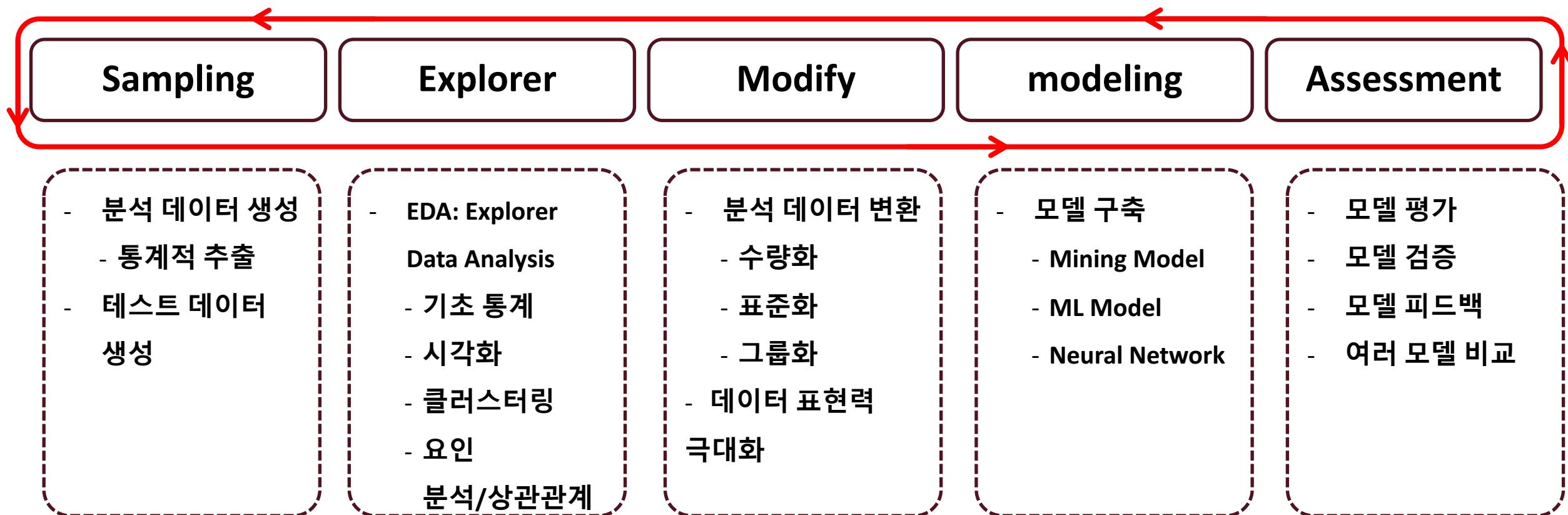
CRISP-DM: 데이터 마이닝 표준



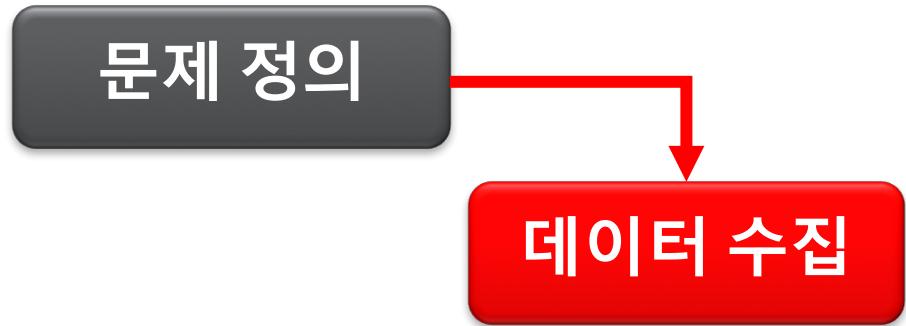
SEMMA: SAS 프로세스



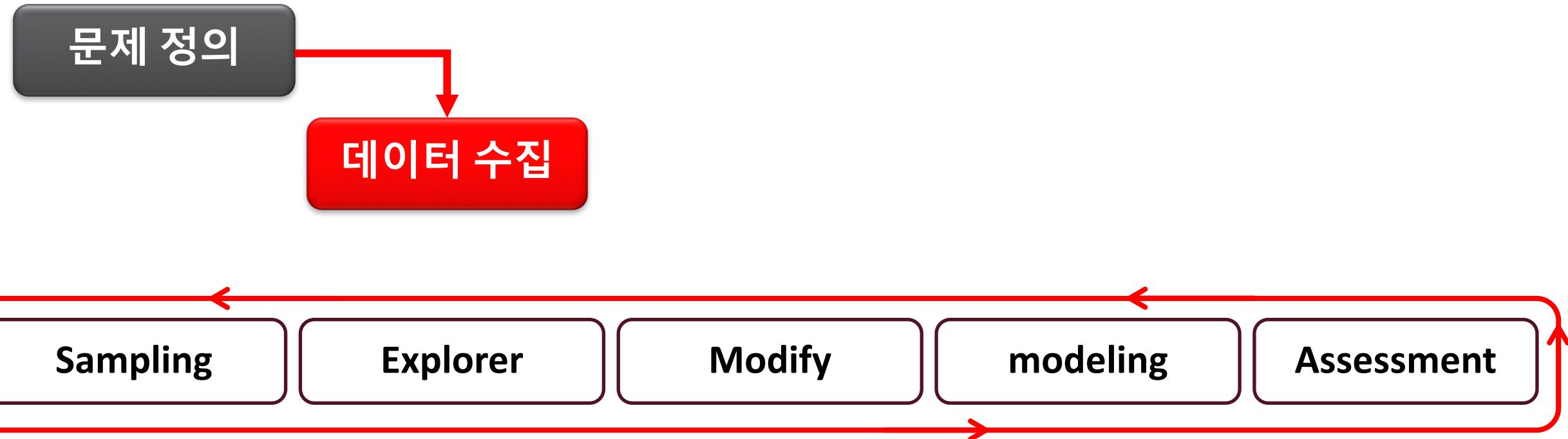
데이터 분석 프로세스: SEMMA



데이터 분석 프로세스



데이터 분석 프로세스



데이터 로딩

```
[2]: import pandas as pd  
boston_df=pd.read_csv("./boston_house_price.csv")
```

```
[4]: boston_df.head()
```

```
[4]:
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

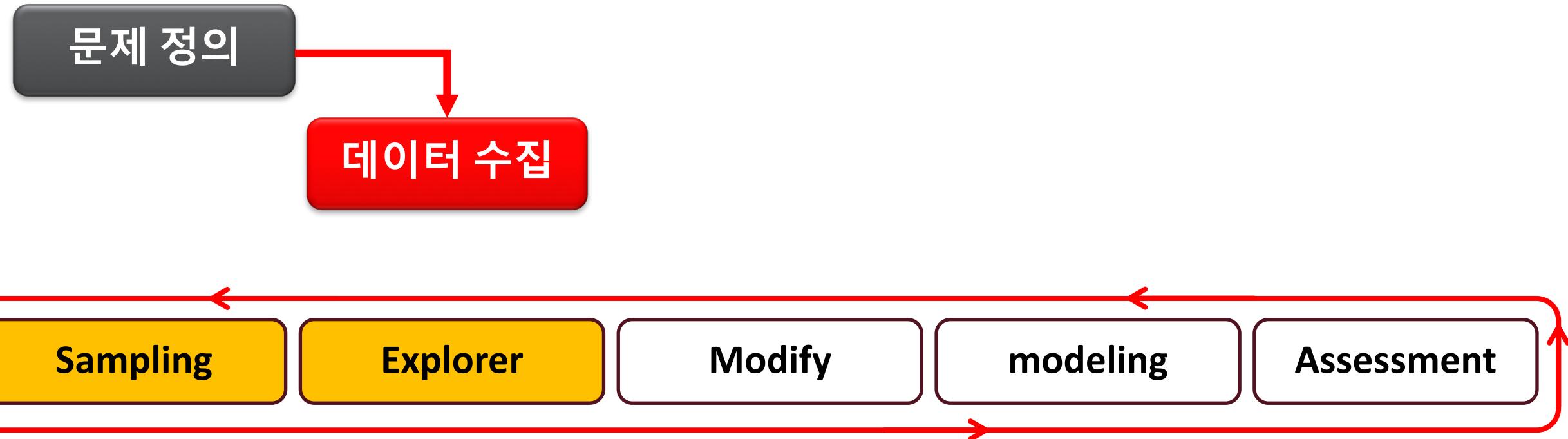
```
[8]: boston_df.shape
```

```
[8]: (506, 14)
```

속성

속성	설명
CRIM	자치시(town) 별 1인당 범죄율
ZN	25,000 평방피트를 초과하는 거주지역의 비율
INDUS	비소매상업지역이 점유하고 있는 토지의 비율
CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
NOX	10ppm 당 농축 일산화질소
RM	주택 1가구당 평균 방의 개수
AGE	1940년 이전에 건축된 소유 주택의 비율
DIS	5개의 보스턴 직업센터까지의 접근성 지수
RAD	방사형 도로까지의 접근성 지수
TAX	10,000 달러 당 재산세율
PTRATIO	자치시(town)별 학생/교사 비율
B	$1000(Bk-0.63)^2$, 여기서 Bk는 자치시 별 흑인의 비율을 말함.
LSTAT	모집단의 하위계층의 비율(%)
MEDV	본인 소유의 주택가격(단위: \$1,000)

데이터 분석 프로세스

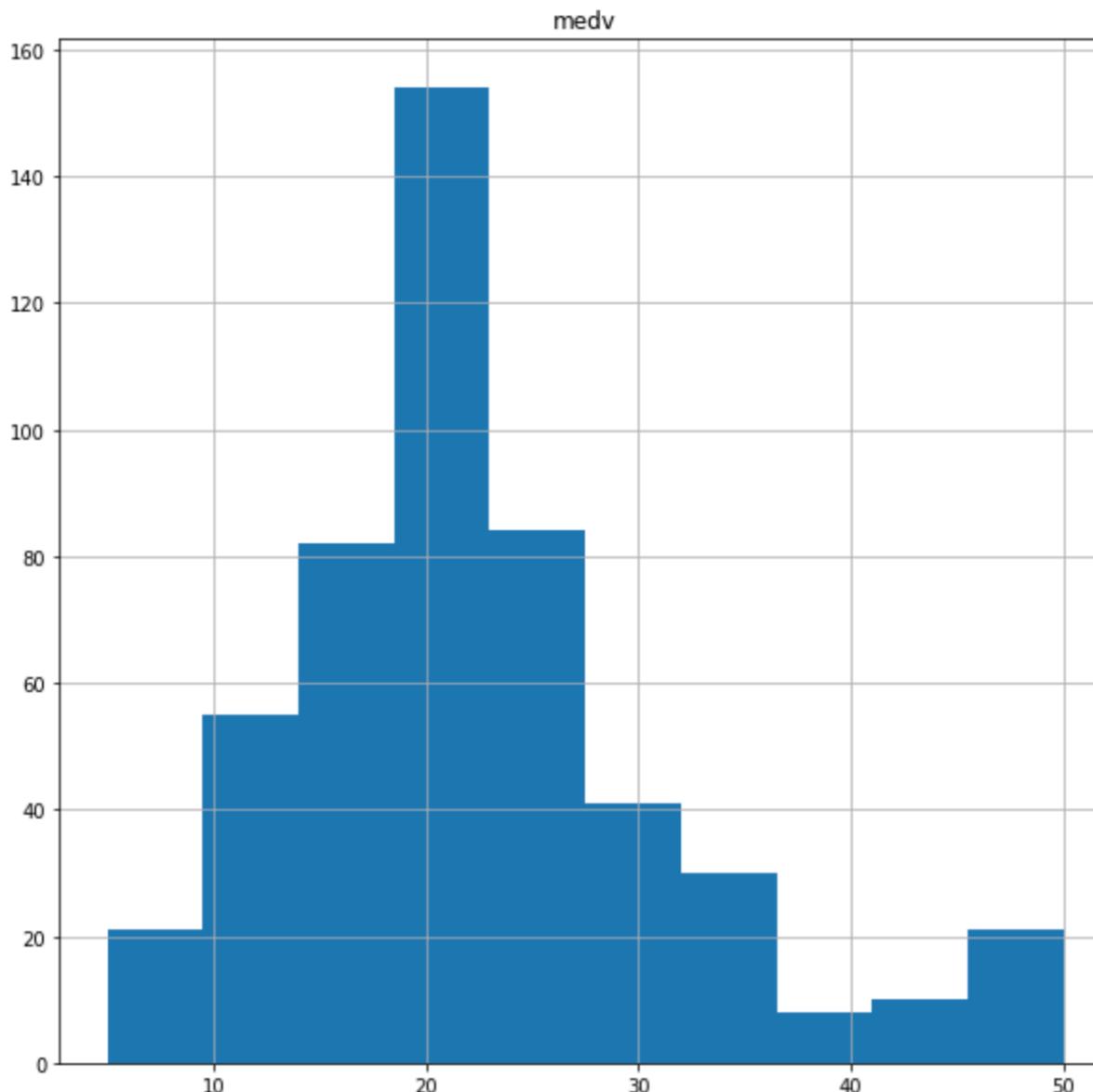


EDA: Explorer Data Analysis

[9]: boston_df.describe()																
	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv		
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063	22.532806		
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104		
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000		
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000	17.025000		
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000		
75%	3.677082	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000		
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000		

```
In [87]: %matplotlib inline  
boston_df.hist("medv", figsize=(10,10))
```

```
Out[87]: array([<matplotlib.axes._subplots.AxesSubplot object at 0x2969e50ac8>],  
               dtype=object)
```



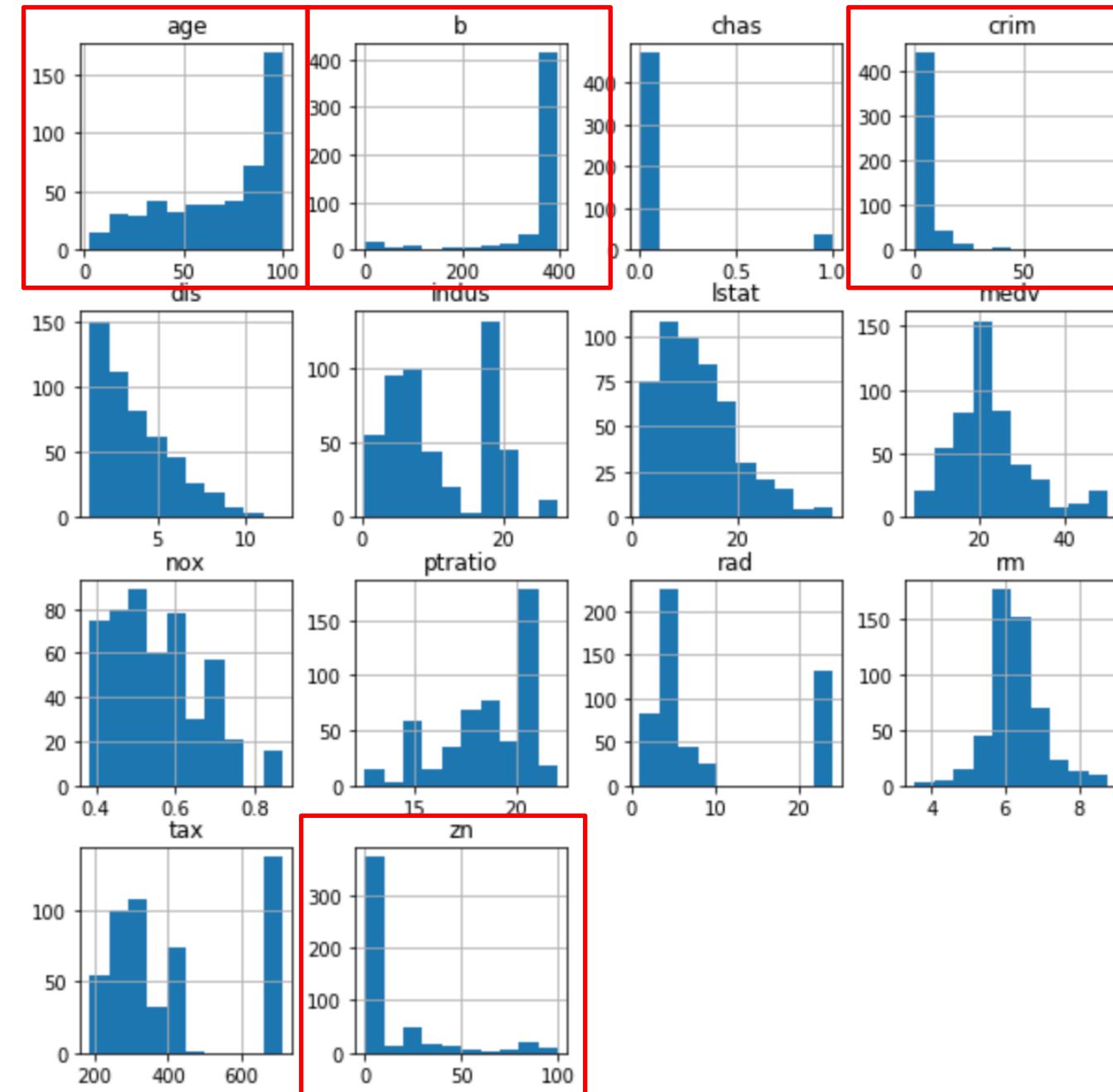
```
boston_df['medv'].describe()
```

count	506.000000
mean	22.532806
std	9.197104
min	5.000000
25%	17.025000
50%	21.200000
75%	25.000000
max	50.000000

Name: medv, dtype: float64

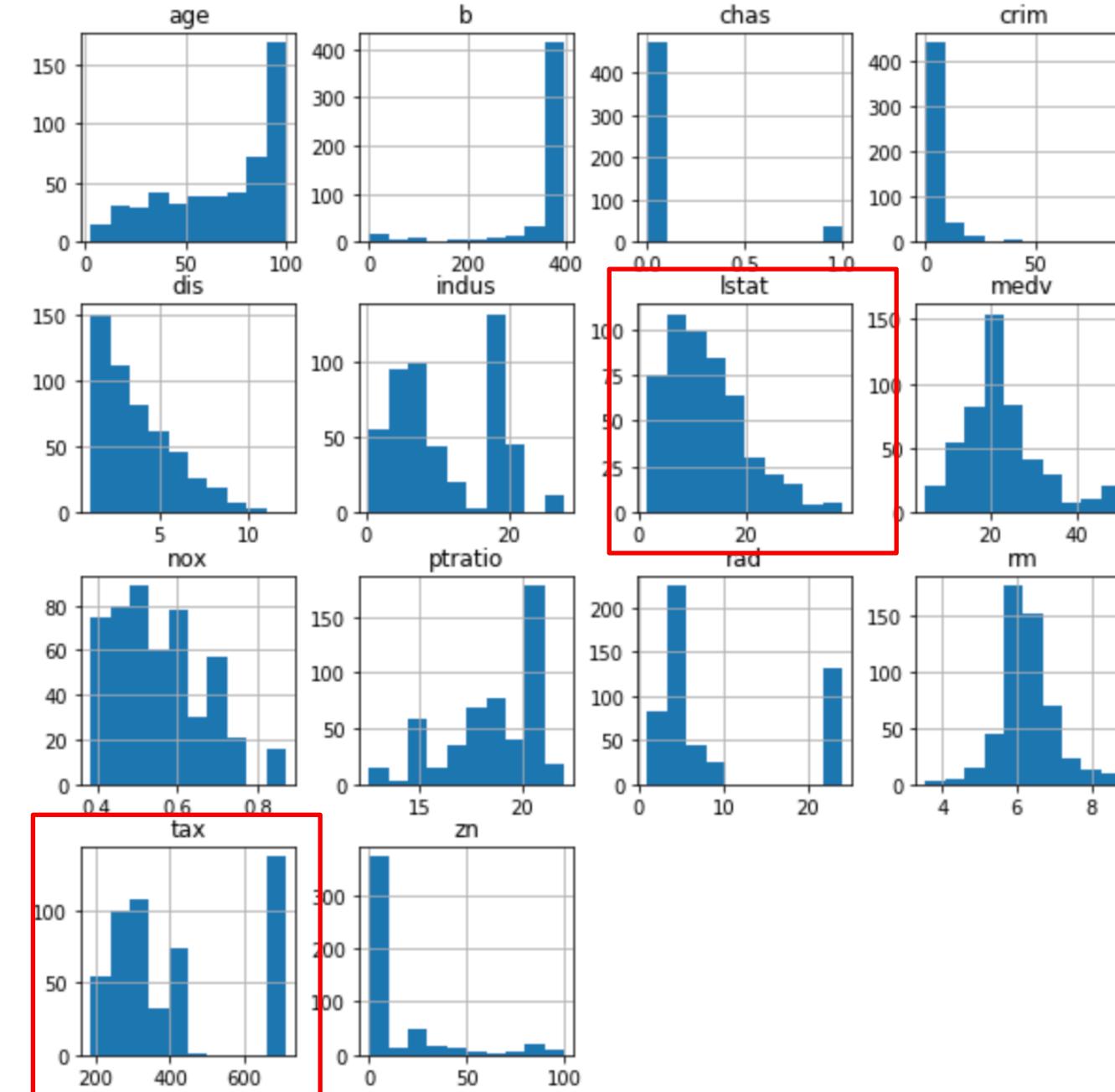
- 보스턴 집값: 정규 분포에서 벗어난 비대칭성
- 보스턴의 집값은 평균적으로 \$22000 (MEDV 평균: 22.5328063)
- 보스턴의 가장 비싼 집값은 \$50,000
- 이 값은 Outlier로 구분, 분석에서 제외 (MEDV가 50인 16개의 ROW 제외)
- Outlier 판단은 IQR 방법을 사용, 계산 값을 넘어서는 값을 Outlier로 판단
 - $IQR = Q3 - Q1$ (50%의 흘어진 정도)

```
hists = boston_df.hist(figsize=(10,10))
```



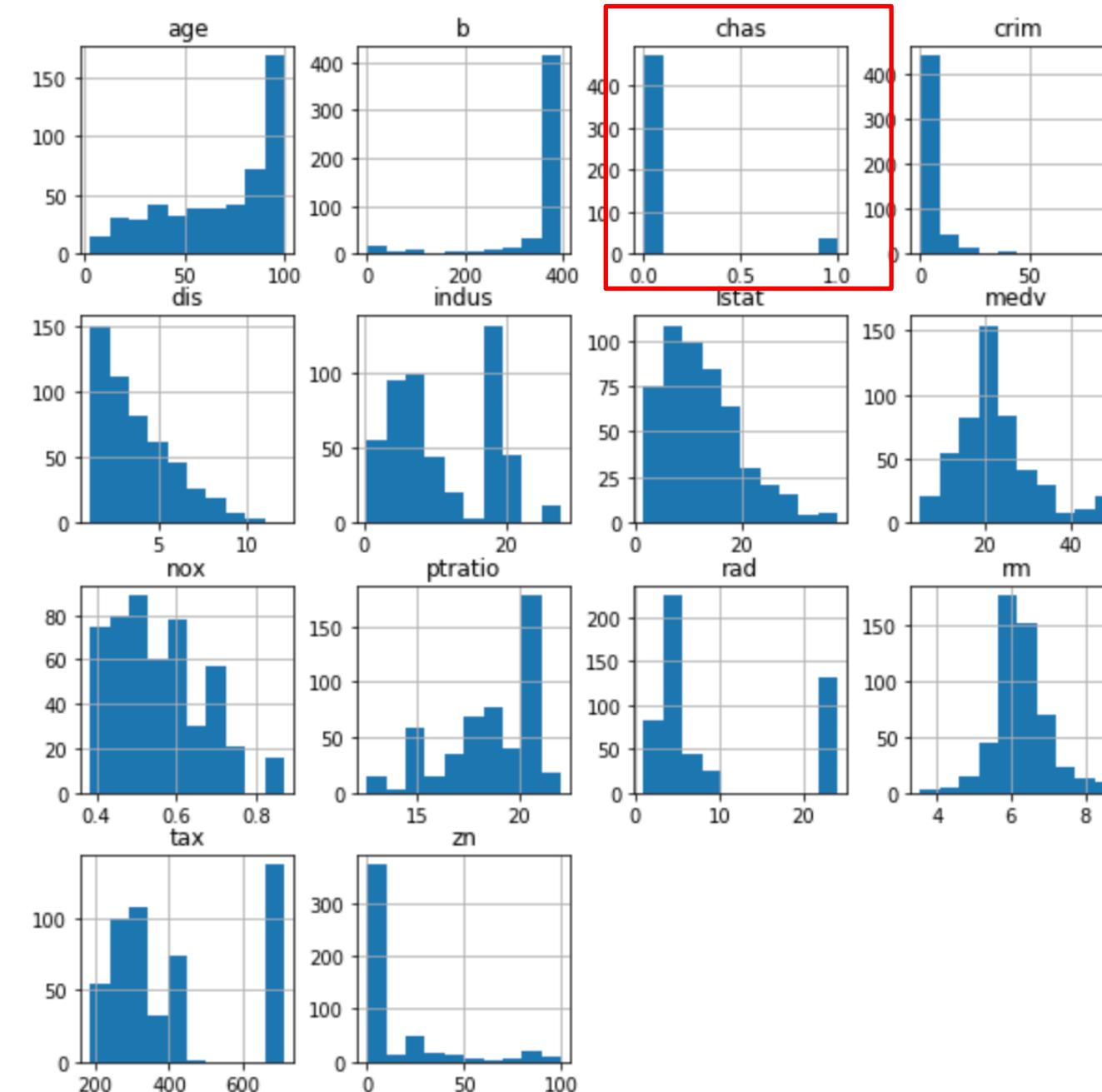
- CRIM, ZN, AGE 및 B와 같은 변수들은
지수 분포를 따르는 변수

```
hists = boston_df.hist(figsize=(10,10))
```



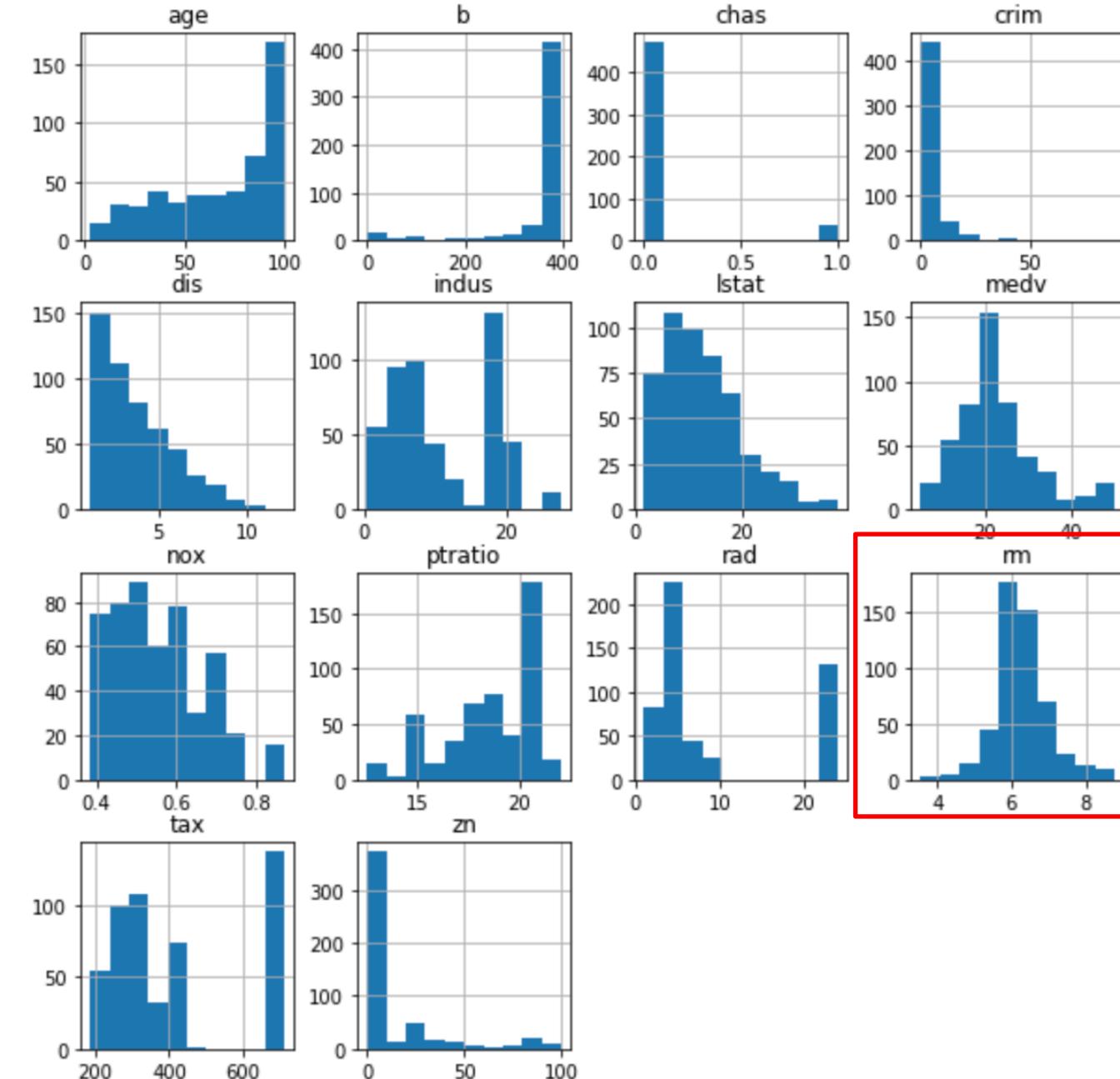
- CRIM, ZN, AGE 및 B와 같은 변수들은 지수 분포를 따르는 변수
- RAD와 TAX 변수는 이분산 분포 형태

```
hists = boston_df.hist(figsize=(10,10))
```



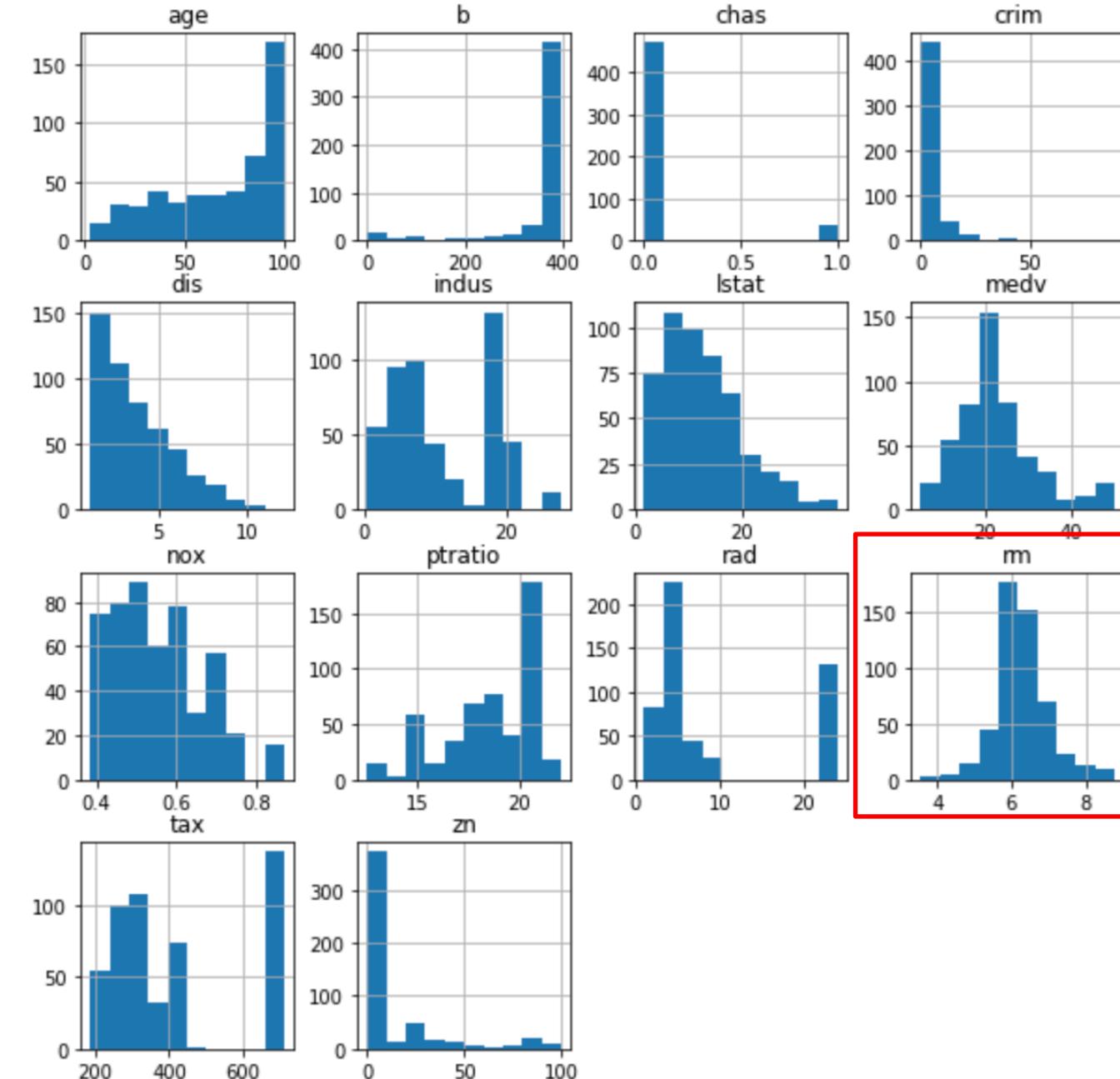
- CRIM, ZN, AGE 및 B와 같은 변수들은 지수 분포를 따르는 변수
- RAD와 TAX 변수는 이분산 분포 형태
- CHAS는 범주형 변수, 회귀에 적당하지 않음

```
hists = boston_df.hist(figsize=(10,10))
```



- CRIM, ZN, AGE 및 B와 같은 변수들은 지수 분포를 따르는 변수
- RAD와 TAX 변수는 이분산 분포 형태
- CHAS는 범주형 변수, 회귀에 적당하지 않음
- RM 정규 분포에 가장 가까움

```
hists = boston_df.hist(figsize=(10,10))
```



- CRIM, ZN, AGE 및 B와 같은 변수들은 지수 분포를 따르는 변수
- RAD와 TAX 변수는 이분산 분포 형태
- CHAS는 범주형 변수, 회귀에 적당하지 않음
- RM 정규 분포에 가장 가까움
 - outlier: 8.7, 3.5 제거?

EDA: 상관계수(Correlation coefficient)

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

EDA: 상관계수(Correlation coefficient)

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

$$\begin{aligned} R &= \begin{bmatrix} r_{x1x1} & r_{x1x2} & \dots & r_{x1xm} \\ r_{x2x1} & r_{x2x2} & \dots & r_{x2xm} \\ \vdots & \vdots & & \vdots \\ r_{xnx1} & r_{xnx2} & \dots & r_{xnxm} \end{bmatrix} \\ &= \frac{1}{n} Z^T Z \end{aligned}$$

Correlation matrix 상관관계가 높은 변수 확인

```
corr = boston_df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

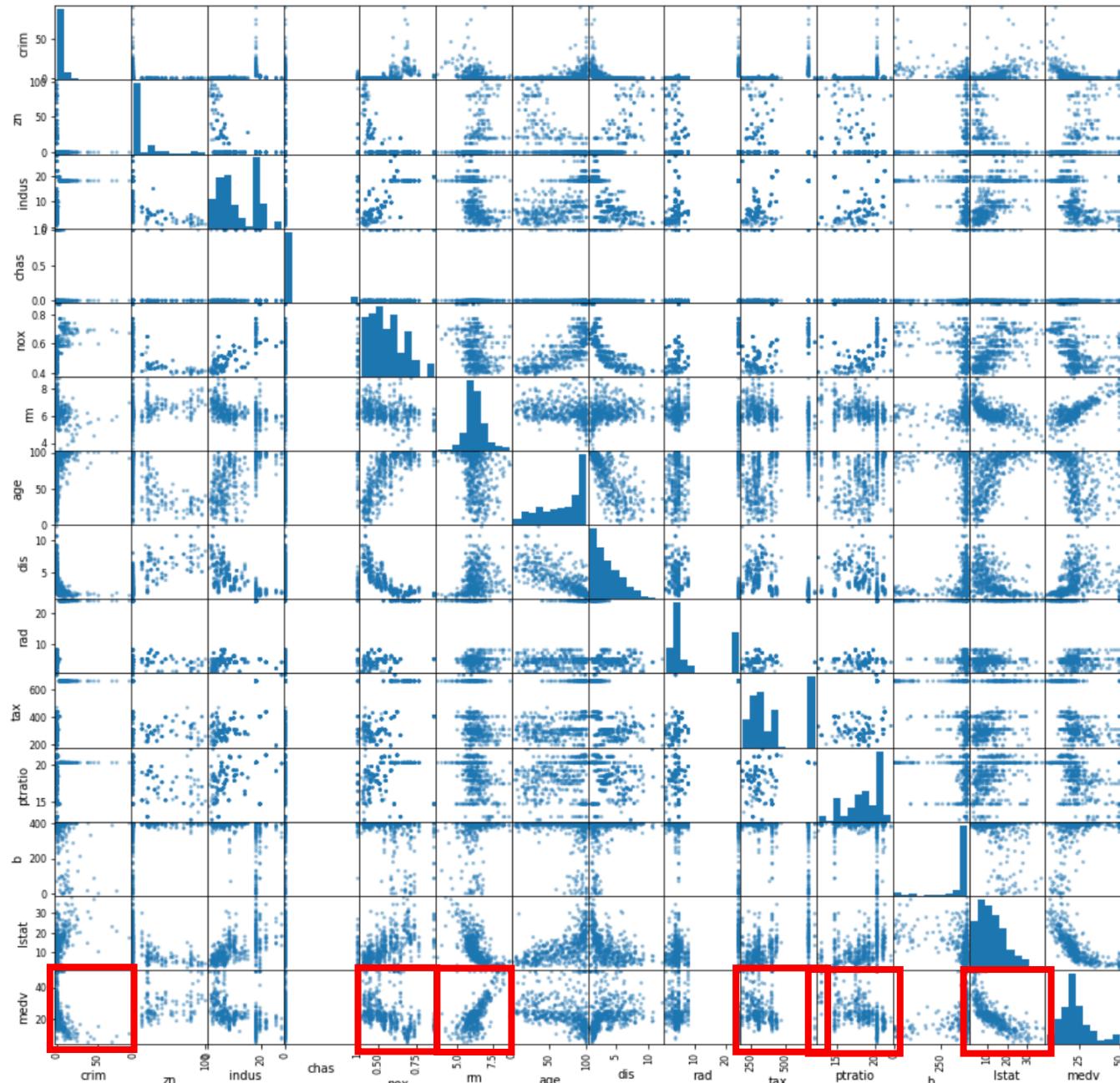
	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	Istat
crim	1	-0.200469	0.406583	-0.0558916	0.420972	-0.219247	0.352734	-0.37967	0.625505	0.582764	0.289946	-0.385064	0.455621
zn	-0.200469	1	-0.533828	-0.0426967	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.17552	-0.412995
indus	0.406583	-0.533828	1	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.72076	0.383248	-0.356977	0.6038
chas	-0.0558916	-0.0426967	0.062938	1	0.0912028	0.0912512	0.0865178	-0.0991758	-0.00736824	-0.0355865	-0.121515	0.0487885	-0.0539293
nox	0.420972	-0.516604	0.763651	0.0912028	1	-0.302188	0.73147	-0.76923	0.611441	0.668023	0.188933	-0.380051	0.590879
rm	-0.219247	0.311991	-0.391676	0.0912512	-0.302188	1	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808
age	0.352734	-0.569537	0.644779	0.0865178	0.73147	-0.240265	1	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339
dis	-0.37967	0.664408	-0.708027	-0.0991758	-0.76923	0.205246	-0.747881	1	-0.494588	-0.534432	-0.232471	0.291512	-0.496996
rad	0.625505	-0.311948	0.595129	-0.00736824	0.611441	-0.209847	0.456022	-0.494588	1	0.910228	0.464741	-0.444413	0.488676
tax	0.582764	-0.314563	0.72076	-0.0355865	0.668023	-0.292048	0.506456	-0.534432	0.910228	1	0.460853	-0.441808	0.543993
ptratio	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1	-0.177383	0.374044
b	-0.385064	0.17552	-0.356977	0.0487885	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177383	1	-0.366087
Istat	0.455621	-0.412995	0.6038	-0.0539293	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366087	1
medv	-0.388305	0.360445	-0.483725	0.17526	-0.427321	0.69536	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.737663

Correlation matrix 상관관계가 높은 변수 확인(LSTAT, PTRATIO, RM, INDUS, TAX, NOX)

```
corr = boston_df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	Istat
crim	1	-0.200469	0.406583	-0.0558916	0.420972	-0.219247	0.352734	-0.37967	0.625505	0.582764	0.289946	-0.385064	0.455621
zn	-0.200469	1	-0.533828	-0.0426967	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.17552	-0.412995
indus	0.406583	-0.533828	1	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.72076	0.383248	-0.356977	0.6038
chas	-0.0558916	-0.0426967	0.062938	1	0.0912028	0.0912512	0.0865178	-0.0991758	-0.00736824	-0.0355865	-0.121515	0.0487885	-0.0539293
nox	0.420972	-0.516604	0.763651	0.0912028	1	-0.302188	0.73147	-0.76923	0.611441	0.668023	0.188933	-0.380051	0.590879
rm	-0.219247	0.311991	-0.391676	0.0912512	-0.302188	1	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808
age	0.352734	-0.569537	0.644779	0.0865178	0.73147	-0.240265	1	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339
dis	-0.37967	0.664408	-0.708027	-0.0991758	-0.76923	0.205246	-0.747881	1	-0.494588	-0.534432	-0.232471	0.291512	-0.496996
rad	0.625505	-0.311948	0.595129	-0.00736824	0.611441	-0.209847	0.456022	-0.494588	1	0.910228	0.464741	-0.444413	0.488676
tax	0.582764	-0.314563	0.72076	-0.0355865	0.668023	-0.292048	0.506456	-0.534432	0.910228	1	0.460853	-0.441808	0.543993
ptratio	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1	-0.177383	0.374044
b	-0.385064	0.17552	-0.356977	0.0487885	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177383	1	-0.366087
Istat	0.455621	-0.412995	0.6038	-0.0539293	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366087	1
medv	-0.388305	0.360445	-0.483725	0.17526	-0.427321	0.69536	0.376955	0.249929	-0.381626	-0.468536	0.507787	0.333461	-0.737663

```
from pandas.plotting import scatter_matrix  
scatter_mx=scatter_matrix(boston_df,figsize=(16,16))
```



Feature Selection

- LSTAT
- PTRATIO
- RM
- INDUS
- TAX
- NOX

속성	설명
CRIM	자치시(town) 별 1인당 범죄율
ZN	25,000 평방피트를 초과하는 거주지역의 비율
INDUS	비소매상업지역이 점유하고 있는 토지의 비율
CHAS	찰스강에 대한 더미변수(강의 경계에 위치한 경우는 1, 아니면 0)
NOX	10ppm 당 농축 일산화질소
RM	주택 1가구당 평균 방의 개수
AGE	1940년 이전에 건축된 소유 주택의 비율
DIS	5개의 보스턴 직업센터까지의 접근성 지수
RAD	방사형 도로까지의 접근성 지수
TAX	10,000 달러 당 재산세율
PTRATIO	자치시(town)별 학생/교사 비율
B	$1000(Bk-0.63)^2$, 여기서 Bk는 자치시 별 특인의 비율을 말함.
LSTAT	모집단의 하위계층의 비율(%)
MEDV	본인 소유의 주택가격(단위: \$1,000)

데이터 분할

```
array = dataset.values
X = array[:,0:6]
Y = array[:,6]
validation_size = 0.20
seed = 7

X_train, X_validation, y_train, y_validation = model_selection.train_test_split(
    X, Y, test_size=validation_size, random_state=seed)
(X_train.shape, X_validation.shape, y_train.shape, y_validation.shape)

((404, 6), (102, 6), (404,), (102,))
```

Model: Linear Regression

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

lr = LinearRegression()
lr.fit(X_train, y_train)

y_predict = lr.predict(X_train)
score = metrics.r2_score(y_train, y_predict)
print("training data R2:", score)
```

```
training data R2: 0.7191347800673977
```

```
y_predict = lr.predict(X_validation)
score = metrics.r2_score(y_validation, y_predict)
print("validation data R2:", score)
```

```
validation data R2: 0.4909674625349041
```

Mean Squire Error =

$$\frac{1}{n} \sum_{i=1}^n (y_i - t_i)^2$$

R Square =

$$1 - \frac{MSE}{VAR(y)}$$

Solution

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

$$3w_1 + 5w_2 - 4w_3 = 7$$

$$-3w_1 - 2w_2 + 4w_3 = 4$$

$$6w_1 - 1w_2 + 8w_3 = 4$$

Solution: 어떤 솔루션이 가장 좋을까?

$$3w_1 + 5w_2 - 4w_3 = 7 \quad (-1, 2, 0)$$

$$-3w_1 - 2w_2 + 4w_3 = 4 \quad (0, 2, 4/3)$$

$$6w_1 - 1w_2 + 8w_3 = 4 \quad (5, 2, 18/4)$$

Model: Lasso – 선형 회귀의 의미 없는 가중치 제거

```
lasso = Lasso()  
lasso.fit(X_train, y_train)  
  
y_predict = lasso.predict(X_train)  
score = metrics.r2_score(y_train, y_predict)  
print("training data, Lasso - R2:", score)  
  
y_predict = lasso.predict(X_validation)  
score = metrics.r2_score(y_validation, y_predict)  
print("validation data, Lasso - R2:", score)  
  
training data, Lasso - R2: 0.6820528235623888  
validation data, Lasso - R2: 0.5756970763637104
```

$$w = \arg \min_w \left(\sum_{i=1}^N e_i^2 + \lambda \sum_{j=1}^M |w_j| \right)$$

미분시 가중치 항은 상수

Model: Ridge – 선형 회귀의 가중치를 작게

```
ridge = Ridge()
ridge.fit(X_train, y_train)

y_predict = ridge.predict(X_train)
score = metrics.r2_score(y_train, y_predict)
print("training data, Ridge - R2:", score)

y_predict = ridge.predict(X_validation)
score = metrics.r2_score(y_validation, y_predict)
print("validation data, Ridge - R2:", score)

training data, Ridge - R2: 0.7190244677684591
validation data, Ridge - R2: 0.49319666826328645
```

$$w = \arg \min_w \left(\sum_{i=1}^N e_i^2 + \lambda \sum_{j=1}^M w_j^2 \right)$$

미분시 가중치 항은 L2 Norm인 Weight 항은 비율이 됨

Model: ElasticNet = Ridge + Lasso

```
from sklearn.linear_model import ElasticNet  
  
en = ElasticNet()  
en.fit(X_train, y_train)  
  
y_predict = en.predict(X_train)  
score = metrics.r2_score(y_train, y_predict)  
print("training data, ElasticNet - R2:", score)  
  
y_predict = en.predict(X_validation)  
score = metrics.r2_score(y_validation, y_predict)  
print("validation data, ElasticNet - R2:", score)
```

```
training data, ElasticNet - R2: 0.6565504044092061  
validation data, ElasticNet - R2: 0.5906796176638256
```

$$w = \arg \min_w \left(\sum_{i=1}^N e_i^2 + \lambda_1 \sum_{j=1}^M |w_j| + \lambda_2 \sum_{j=1}^M w_j^2 \right)$$

Feature Extraction

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
poly.fit(X)
x_data = poly.transform(X)

X_train, X_validation, y_train, y_validation = model_selection.train_test_split(
    x_data, Y, test_size=validation_size, random_state=seed)
(X_train.shape, X_validation.shape, y_train.shape, y_validation.shape)

((404, 28), (102, 28), (404,), (102,))

lr = LinearRegression()
lr.fit(X_train, y_train)

y_predict = lr.predict(X_train)
score = metrics.r2_score(y_train, y_predict)
print("training data Linear Regression - R2:", score)

y_predict = lr.predict(X_validation)
score = metrics.r2_score(y_validation, y_predict)
print("validation data Linear Regression - R2:", score)

training data Linear Regression - R2: 0.8748489620171976
validation data Linear Regression - R2: 0.745611172233344
```

Model 검증

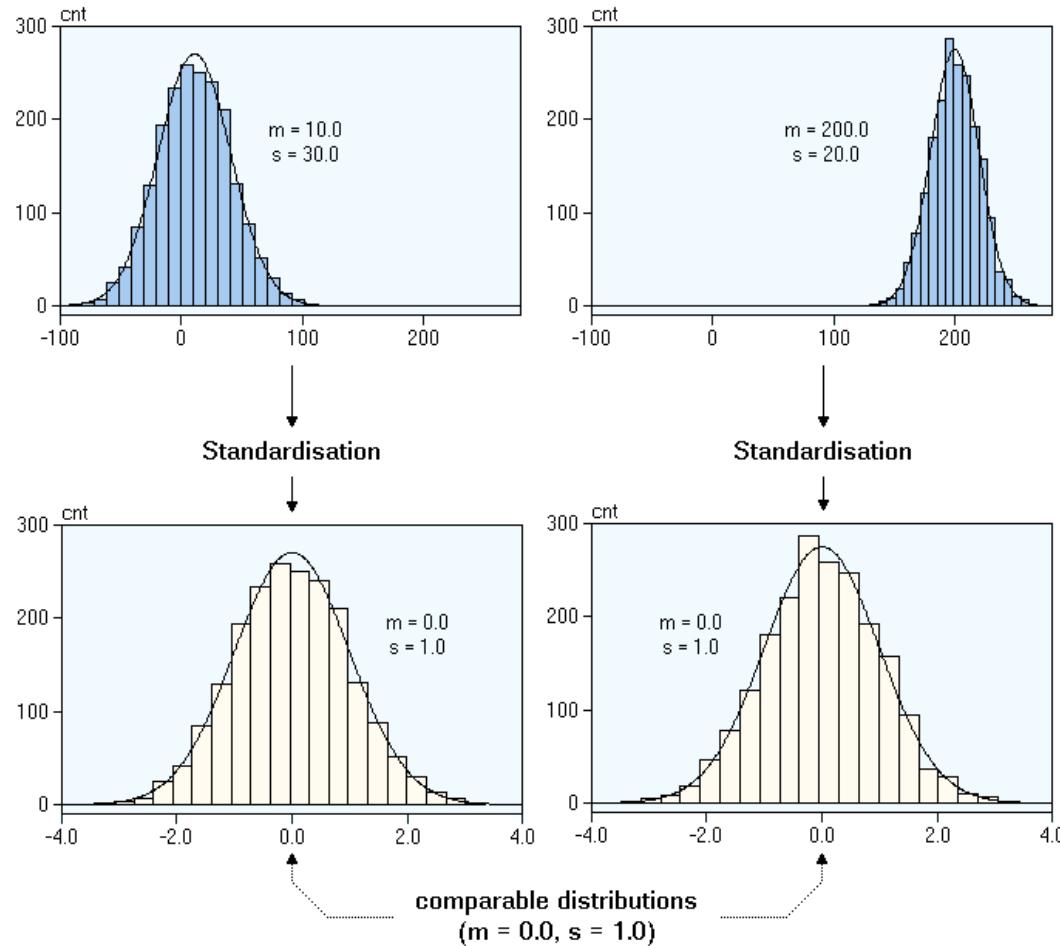
```
seed=35
kfold=KFold(n_splits=10, random_state=seed)
num_folds = 10

models = []
models.append(('LR', LinearRegression()))
models.append(('LASSO', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('KNN', KNeighborsRegressor()))
models.append(('CART', DecisionTreeRegressor()))
models.append(('SVR', SVR()))

results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=num_folds, random_state=seed)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='r2')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

LR: 0.834867 (0.076225)
LASSO: 0.830806 (0.084568)
EN: 0.831369 (0.084819)
KNN: 0.694156 (0.115922)
```

Standardization



$$z = \frac{x_i - \mu}{\sigma}$$

Model 검증: 표준화 데이터

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
pipelines = []
pipelines.append(('ScaledLR', Pipeline([('Scaler', StandardScaler()), ('LR', LinearRegression())])))
pipelines.append(('ScaledLASSO', Pipeline([('Scaler', StandardScaler()), ('LASSO', Lasso())])))
pipelines.append(('ScaledEN', Pipeline([('Scaler', StandardScaler()), ('EN', ElasticNet())])))
pipelines.append(('ScaledKNN', Pipeline([('Scaler', StandardScaler()), ('KNN', KNeighborsRegressor())])))
pipelines.append(('ScaledCART', Pipeline([('Scaler', StandardScaler()), ('CART', DecisionTreeRegressor())])))
pipelines.append(('ScaledSVR', Pipeline([('Scaler', StandardScaler()), ('SVR', SVR())])))
results = []
names = []
for name, model in pipelines:
    kfold = KFold(n_splits=num_folds, random_state=seed)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='r2')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

ScaledLR: 0.834189 (0.077067)
ScaledLASSO: 0.727223 (0.064642)
ScaledEN: 0.702851 (0.062363)
ScaledKNN: 0.823836 (0.073393)
ScaledCART: 0.705543 (0.161044)
ScaledSVR: 0.718610 (0.114292)
```

ADW: 마음은 급한데~~(2019.08.20 03:00)

ORACLE® 클라우드 기반 구조

알 수 없는 애플리케이션 오류

Database name:
DWLEEDAW

Oracle Machine Learning Database 사용자 인증서로 사인인

사용자 이름 *

oml

비밀번호 *

.....

[이용약관 및 개인정보 보호정책](#) [피드백 전송](#)

Copyright (c) 2018 , Oracle and/or its affiliates. All rights reserved.

ADW: 상관 관계

```
BEGIN
  DBMS_PREDICTIVE_ANALYTICS.EXPLAIN(
    data_table_name => 'boston_house',
    explain_column_name => 'MEDV',
    result_table_name => 'ai_explain_output');
END;
End;
```

PL/SQL procedure successfully completed.

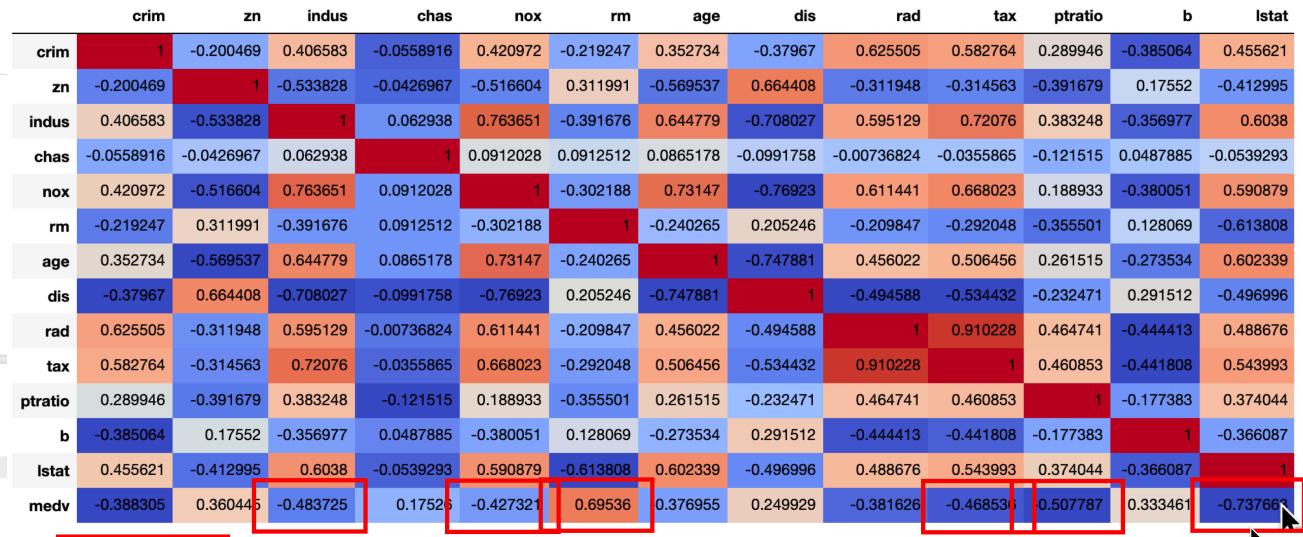
Took 16 sec. Last updated by OML at August 20 2019, 9:04:25 AM.

```
%sql
SELECT attribute_name, round(explanatory_value,4), rank
FROM ai_explain_output
ORDER BY rank, attribute_name;
```



ATTRIBUTE_NAME	ROUND(EXPLANATORY_VALUE,4)	RANK
LSTAT	0.08	1
SEQ	0.0695	2
RM	0.0483	3
INDUS	0.0443	4
NOX	0.0441	5
TAX	0.0347	6
PTRATIO	0.0331	7
CRIM	0.0267	8
RAD	0.017	9

Took 1 sec. Last updated by OML at August 12 2019, 1:20:02 PM.



LSTAT, PTRATIO, RM, INDUS, TAX, NOX

ADW: 데이터 분할

```
%script
/* Split the Data into N1_TRAIN_DATA and N1_TEST_DATA */
BEGIN
EXECUTE IMMEDIATE 'CREATE OR REPLACE VIEW AS SELECT * FROM boston_house SAMPLE (75) SEED
(1)';
DBMS_OUTPUT.PUT_LINE ('Created N1_TRAIN_DATA');
EXECUTE IMMEDIATE 'CREATE OR REPLACE VIEW N1_TEST_DATA AS SELECT * FROM boston_house
MINUS SELECT * FROM N1_TRAIN_DATA';
DBMS_OUTPUT.PUT_LINE ('Created N1_TEST_DATA');
END;
/
Created N1_TRAIN_DATA
Created N1_TEST_DATA
PL/SQL procedure successfully completed.
-----
Took 0 sec. Last updated by OML at August 05 2019, 4:32:45 PM. (c)
```

FINISHED ▶ ✎ 📄 ⚙

Sklearn Code: 학습 / 검증 데이터 분할

```
array = dataset.values
X = array[:,0:6]
Y = array[:,6]
validation_size = 0.20
seed = 7

X_train, X_validation, y_train, y_validation = model_selection.train_test_split(
    X, Y, test_size=validation_size, random_state=seed)
(X_train.shape, X_validation.shape, y_train.shape, y_validation.shape)

((404, 6), (102, 6), (404,), (102,))
```

ADW: 모델 설정

```
%sql  
CREATE TABLE glmr_BH_settings (  
    setting_name  VARCHAR2(30),  
    setting_value VARCHAR2(4000));  
  
Updated 0 row(s).  
  
Took 0 sec. Last updated by OML at August 12 2019, 1:30:22 PM.
```

```
%script  
BEGIN  
/* Populate settings table */  
INSERT INTO glmr_BH_settings (setting_name, setting_value) VALUES      (dbms_data_mining.algo_name,  
 dbms_data_mining.algo_generalized_linear_model);  
-- output row diagnostic statistics into a table named GLMC_SH_SAMPLE_DIAG  
INSERT INTO glmr_BH_settings (setting_name, setting_value) VALUES      (dbms_data_mining  
 .glms_diagnostics_table_name, 'glmr_BH_diag');  
INSERT INTO glmr_BH_settings (setting_name, setting_value) VALUES      (dbms_data_mining.prep_auto  
 , dbms_data_mining.prep_auto_on);  
-- turn on feature selection  
INSERT INTO glmr_BH_settings (setting_name, setting_value) VALUES      (dbms_data_mining  
 .glms_ftr_selection, dbms_data_mining.glms_ftr_selection_enable);  
-- turn on feature generation  
INSERT INTO glmr_BH_settings (setting_name, setting_value) VALUES      (dbms_data_mining  
 .glms_ftr_generation, dbms_data_mining.glms_ftr_generation_enable);  
--특정 컬럼 가중치를 줄 때  
--INSERT INTO glmr_BH_settings (setting_name, setting_value) VALUES      (dbms_data_mining  
 .odms_row_weight_column_name , '컬럼명' ) -- 숫자타입 컬럼 만 가능함  
--없는 값 처리에 대한 옵션 , 기본 값은 평균  
--INSERT INTO glmr_BH_settings (setting_name, setting_value) VALUES      (dbms_data_mining  
 .odms_missing_value_treatment, dbms_data_mining.odms_missing_value_delete_row);  
-- 리지 적용 L2 노름 적용  
--INSERT INTO glmr_BH_settings (setting_name, setting_value) VALUES      (dbms_data_mining  
 .glms_ridge_regression, dbms_data_mining.glms_ridge_reg_enable);  
--INSERT INTO glmr_BH_settings (setting_name, setting_value) VALUES      (dbms_data_mining  
 .GLMS_VIF_FOR RIDGE, dbms_data_mining.GLMS_VIF_RIDGE_ENABLE);
```

```
class sklearn.cluster.KMeans(  
    n_clusters=8, init='kmeans++', n_init=  
    10, max_iter=300, tol=0.0001, precomp  
    ute_distances='auto', verbose=0, rand  
    om_state=None, copy_x=True, n_jobs  
    =None, algorithm='auto')
```

ADW: 모델 생성

```
%script

declare
  v_xlst dbms_data_mining_transform.TRANSFORM_LIST;

BEGIN
  DBMS_DATA_MINING.CREATE_MODEL(
    model_name          => 'GLMR_BH_Regr',
    mining_function     => dbms_data_mining.regression,
    data_table_name     => 'N1_TRAIN_DATA',
    case_id_column_name => 'SEQ',
    target_column_name  => 'MEDV',
    settings_table_name => 'glmr_BH_settings',
    xform_list          => v_xlst);
END;
```

PL/SQL procedure successfully completed.

```
lr = LinearRegression()
lr.fit(X_train, y_train)

y_predict = lr.predict(X_train)
score = metrics.r2_score(y_train, y_predict)
print("training data Linear Regression - R2:", score)

y_predict = lr.predict(X_validation)
score = metrics.r2_score(y_validation, y_predict)
print("validation data Linear Regression - R2:", score)
```

ADW: 모델 평가

```
%sql
select name, numeric_value, string_value from DM$VGGLMR_BH_Regr
ORDER BY name;
```

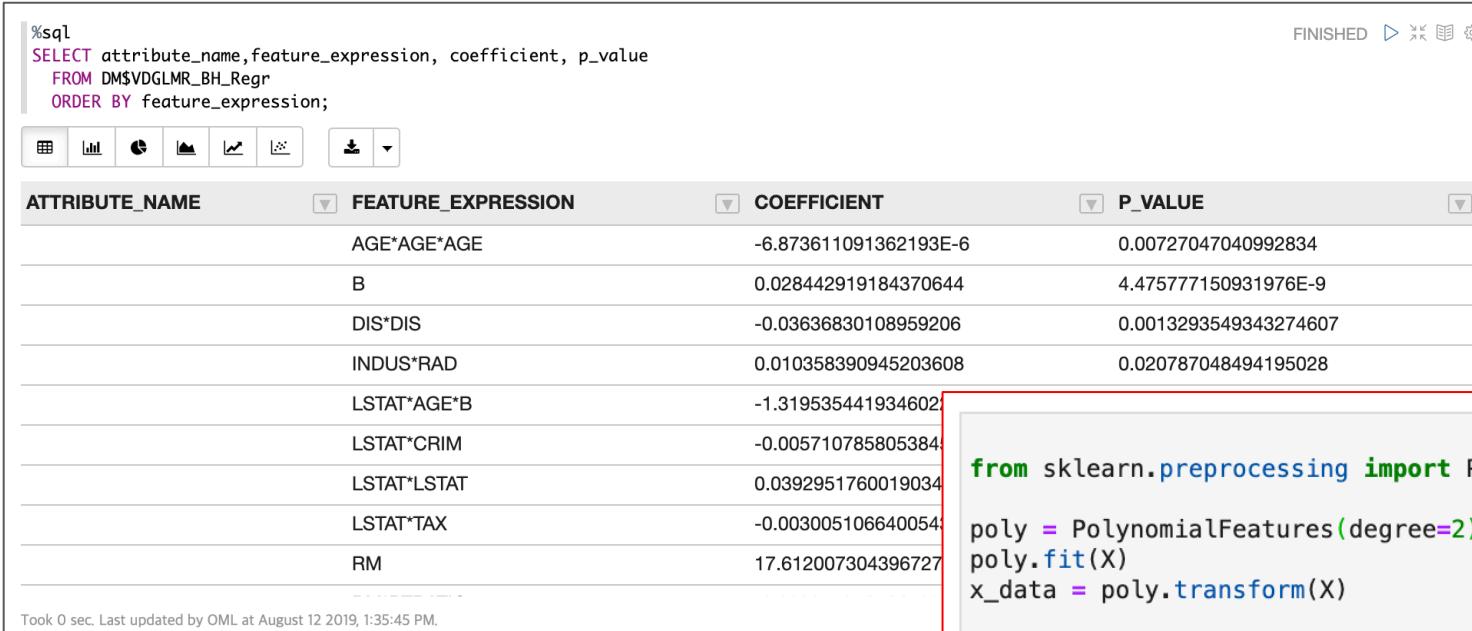
FINISHED ▶ ✎ 📄 ⚙

NAME NUMERIC_VALUE STRING_VALUE

MODEL_F_P_VALUE	0	
MODEL_MEAN_SQUARE	2012.2375015438884	
MODEL_SUM_SQUARES	28171.325021614437	
NUM_PARAMS	15	
NUM_ROWS	367	
ROOT_MEAN_SQ	3.417560090198168	
R_SQ	0.87264773489594027	
SBIC	975.29759537945347	
VALID_COVARIANCE_MATRIX		YES

ADW: Feature Engineering

```
INSERT INTO glmr_BH_settings (setting_name, setting_value) VALUES      (dbms_data_mining.prep_auto
, dbms_data_mining.prep_auto_on);
-- turn on feature selection
INSERT INTO glmr_BH_settings (setting_name, setting_value) VALUES      (dbms_data_mining
.glms_ftr_selection, dbms_data_mining.glms_ftr_selection_enable);
```



The screenshot shows a SQL query results window in Oracle Database SQL Developer. The query selects attribute names, feature expressions, coefficients, and p-values from the DM\$VDGLMR_BH_Regr table. The results are displayed in a grid with columns: ATTRIBUTE_NAME, FEATURE_EXPRESSION, COEFFICIENT, and P_VALUE. The data includes various polynomial terms like AGE*AGE*AGE, B, DIS*DIS, INDUS*RAD, etc., along with their corresponding coefficients and p-values.

ATTRIBUTE_NAME	FEATURE_EXPRESSION	COEFFICIENT	P_VALUE
	AGE*AGE*AGE	-6.873611091362193E-6	0.00727047040992834
	B	0.028442919184370644	4.475777150931976E-9
	DIS*DIS	-0.03636830108959206	0.0013293549343274607
	INDUS*RAD	0.010358390945203608	0.020787048494195028
	LSTAT*AGE*B	-1.319535441934602	
	LSTAT*CRIM	-0.005710785805384	
	LSTAT*LSTAT	0.0392951760019034	
	LSTAT*TAX	-0.003005106640054	
	RM	17.612007304396727	

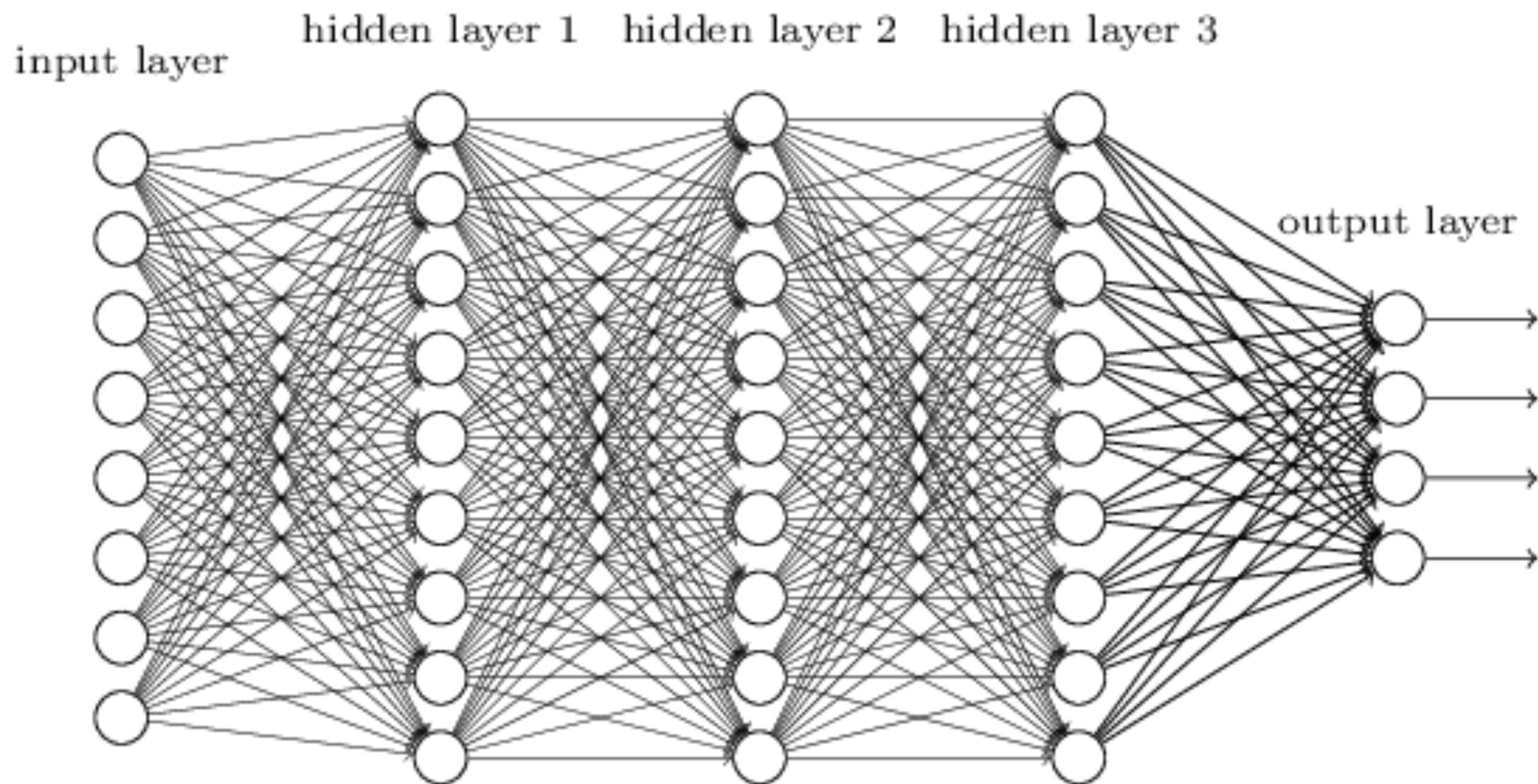
Took 0 sec. Last updated by OML at August 12 2019, 1:35:45 PM.

Feature Extraction with Sklearn

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2)
poly.fit(X)
x_data = poly.transform(X)

X_train, X_validation, y_train, y_validation = model_selection.train_test_split(
    x_data, Y, test_size=validation_size, random_state=seed)
(X_train.shape, X_validation.shape, y_train.shape, y_validation.shape)

((404, 28), (102, 28), (404,), (102,))
```





+ 코드 + 텍스트

다시 연결

수정 가능



목차

코드 스니펫

파일



Filter code snippets

Adding form fields →

Camera Capture →

Cross-output communication →

display.Javascript to execute JavaScript f... →

Downloading files or importing data from... →

Downloading files to your local file system →

Evaluate a Javascript expression from Py... →

Adding form fields

삽입

Forms example

Forms support multiple types of fields with type checking including sliders, date pickers, input fields, dropdown menus, and dropdown menus that allow input.

Drive Mount

```
[ ] from google.colab import drive  
drive.mount('/gdrive')
```

↳ Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=00000000000000000000000000000000&redirect_uri=https%3A%2F%2Faccounts.google.com%2Foauth2%2Fcallback&response_type=code&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%2Bfile

Enter your authorization code:
.....

Mounted at /gdrive

Drive already mounted at /gdrive; to attempt to forcibly remount, call drive.remount()

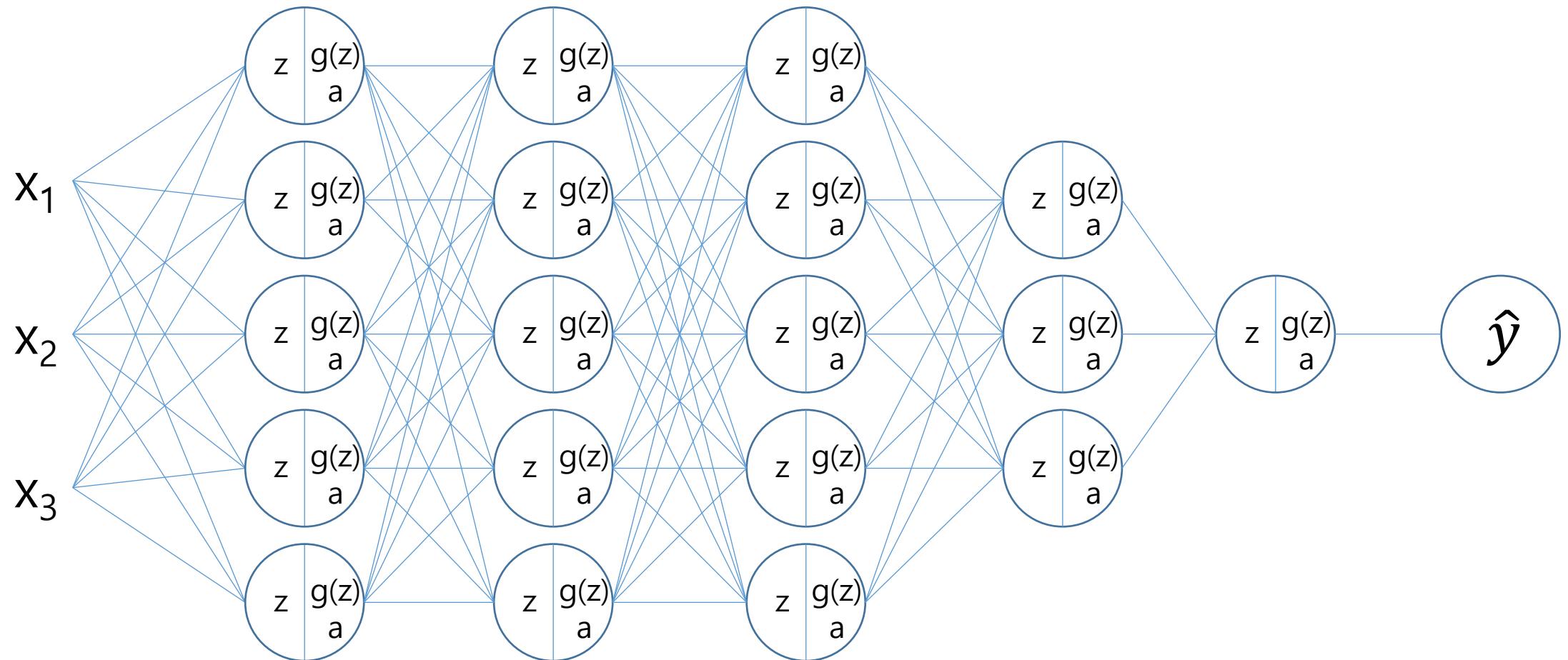
```
[ ] import pandas as pd  
import numpy as np  
from sklearn.model_selection import train_test_split  
from sklearn import preprocessing
```

```
[ ] boston_df = pd.read_csv("/gdrive/My Drive/boston_house_price.csv")
```

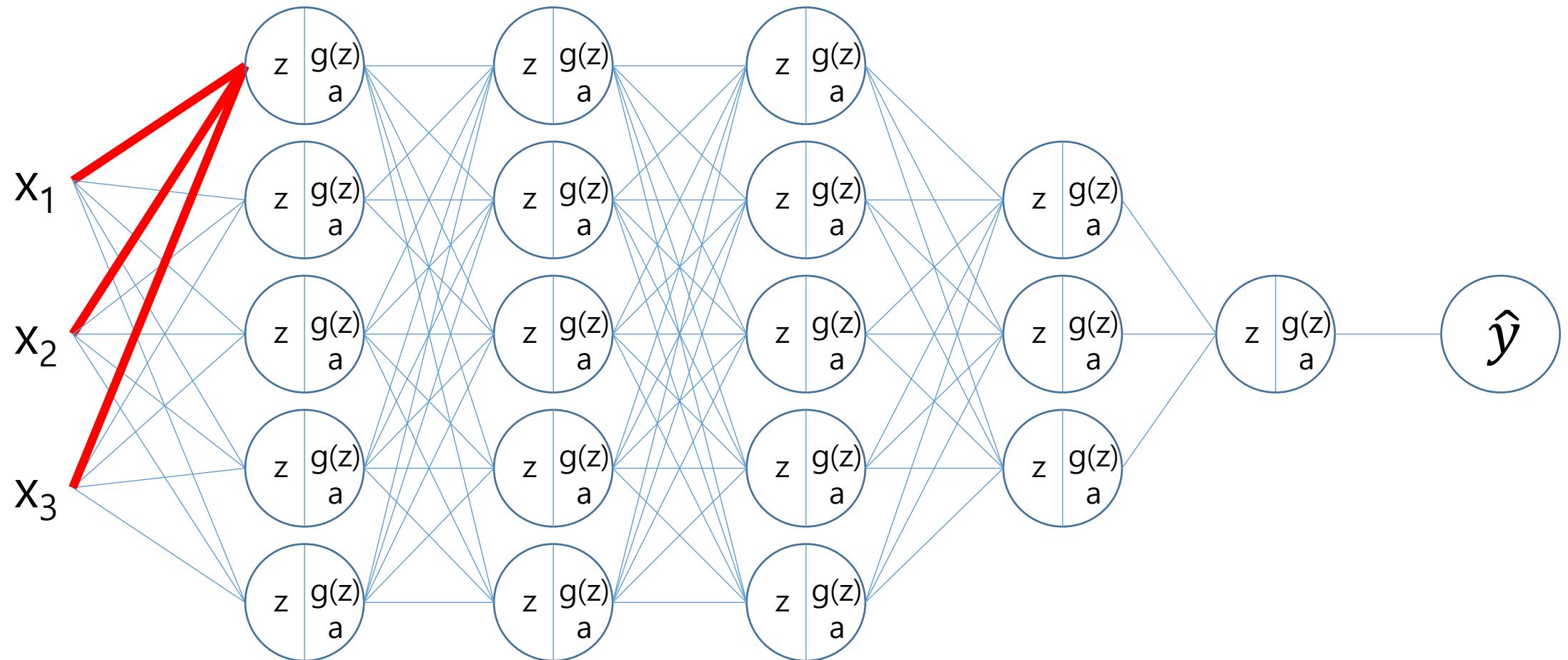
```
[ ] boston_df.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90

Deep Learning: Linear Regression



Deep Learning: Linear Regression



Neural Network Model for Boston House Pricing

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class Regression_NN(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(13, 13)
        self.fc2 = nn.Linear(13, 13)
        self.fc3 = nn.Linear(13, 1)

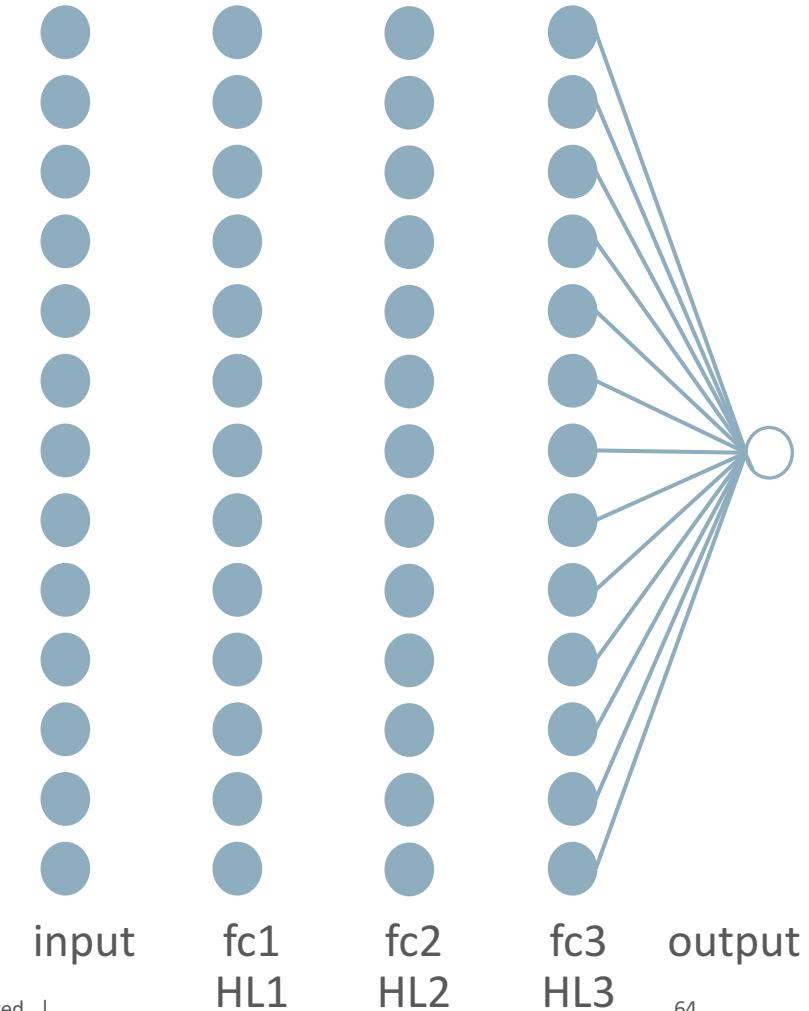
        self.dropout = nn.Dropout(p=0.1)

    def forward(self, x):
        x = self.dropout(F.relu(self.fc1(x)))
        x = self.dropout(F.relu(self.fc2(x)))
#        x = F.relu(self.fc1(x))
#        x = F.relu(self.fc2(x))
        x = self.fc3(x)

        return x

[31] net
```

```
↳ Regression_NN(
  (fc1): Linear(in_features=13, out_features=13, bias=True)
  (fc2): Linear(in_features=13, out_features=13, bias=True)
  (fc3): Linear(in_features=13, out_features=1, bias=True)
  (dropout): Dropout(p=0.1)
)
```



Neural Network Model for Boston House Pricing

```
[35] from sklearn.utils import shuffle
     from torch.autograd import Variable

batch_size = 50
num_epochs = 16000
learning_rate = 0.01

running_loss = 0.0

train_errors = []
test_errors = []

for epoch in range(num_epochs):
    net.train()
    X_train, y_train = shuffle(X_train, y_train)
    inputs = Variable(torch.FloatTensor(X_train))
    labels = Variable(torch.FloatTensor(y_train))
    optimizer.zero_grad()

    outputs = net.forward(inputs)
    loss = criterion(outputs, torch.unsqueeze(labels, dim=1))
    loss.backward()
    optimizer.step()

    train_errors.append(loss.item())

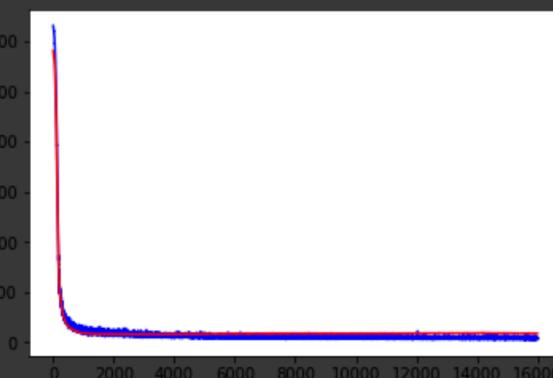
    if epoch%2000 == 0:
        print('Epoch {} loss: {}'.format(epoch+1, loss.item()))

    net.eval()
    test_inputs = Variable(torch.FloatTensor(X_test))
    test_labels = Variable(torch.FloatTensor(y_test))
    test_output = net.forward(test_inputs)
    test_loss = criterion(test_output, torch.unsqueeze(test_labels, dim=1))
    test_errors.append(test_loss.item())
```

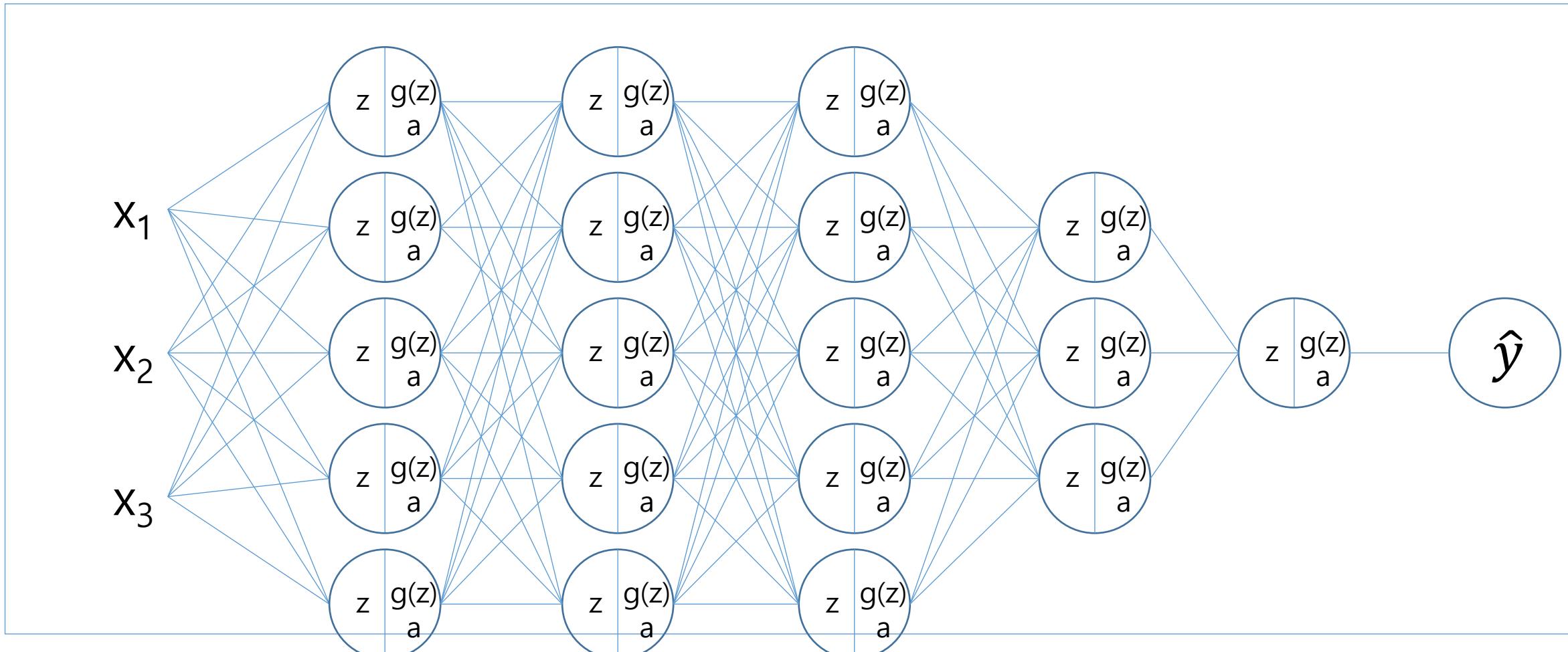
```
↳ Epoch 1 loss: 630.86083984375
Epoch 2001 loss: 14.677943229675293
Epoch 4001 loss: 14.323848724365234
Epoch 6001 loss: 13.816299438476562
Epoch 8001 loss: 8.914133071899414
Epoch 10001 loss: 8.742273330688477
Epoch 12001 loss: 9.113798141479492
Epoch 14001 loss: 8.90151309967041
```

```
[36] import matplotlib.pyplot as plt
     plt.plot(train_errors, 'b-')
     plt.plot(test_errors, 'r-')
```

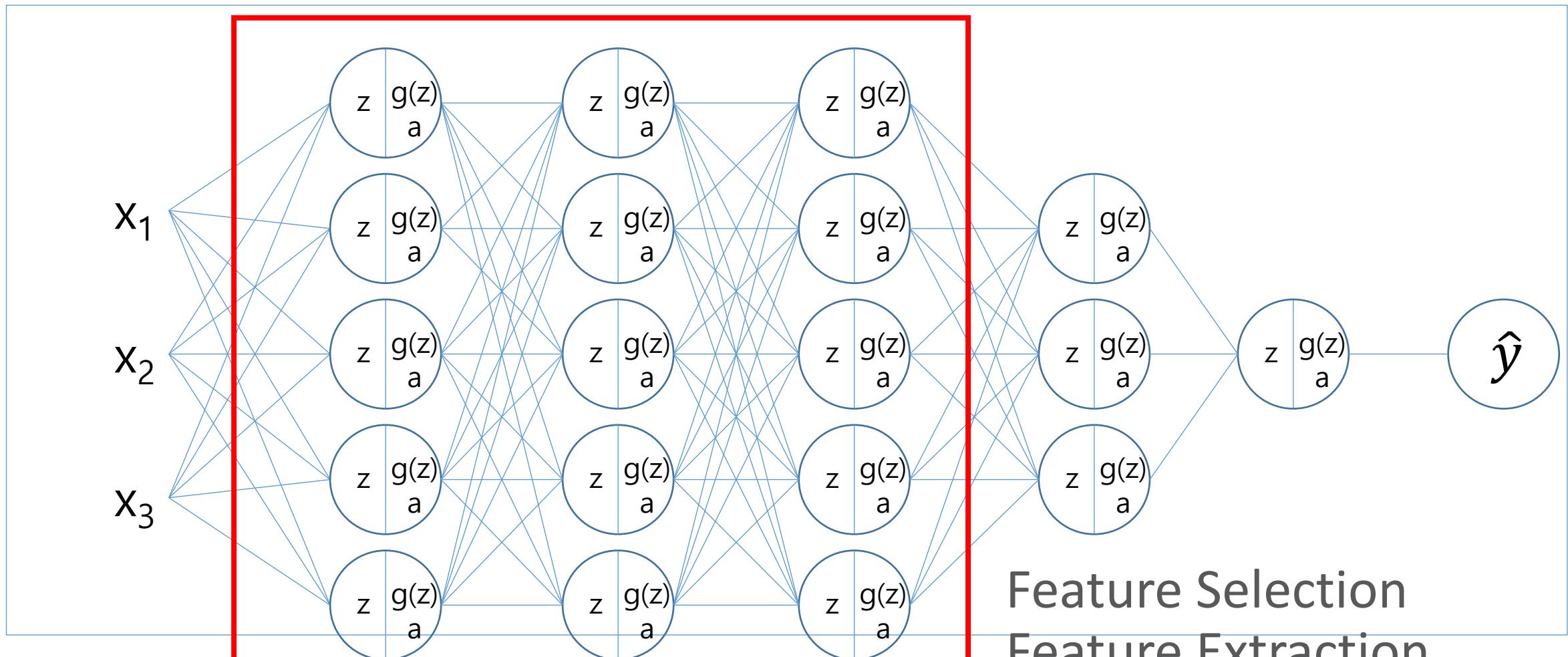
```
↳ [<matplotlib.lines.Line2D at 0x7fd3c5359358>]
```



Deep Learning: Linear Regression



Deep Learning: Linear Regression



사실은....

```
[ ] import pandas as pd
from sklearn.metrics import r2_score
net.eval()
X = Variable(torch.FloatTensor(X_train))
result = net(X)
pred=result.data[:,0].numpy()
print(len(pred),len(y_train))
r2_score(pred,y_train)
```

```
↳ 151 151
0.9595044639121458
```

```
[ ] X = Variable(torch.FloatTensor(X_test))
result = net(X)
pred=result.data[:,0].numpy()
print(len(pred),len(y_test))
r2_score(pred,y_test)
```

```
↳ 355 355
0.7053947564371497
```



Linear Regression → Linear Classification

벡터화

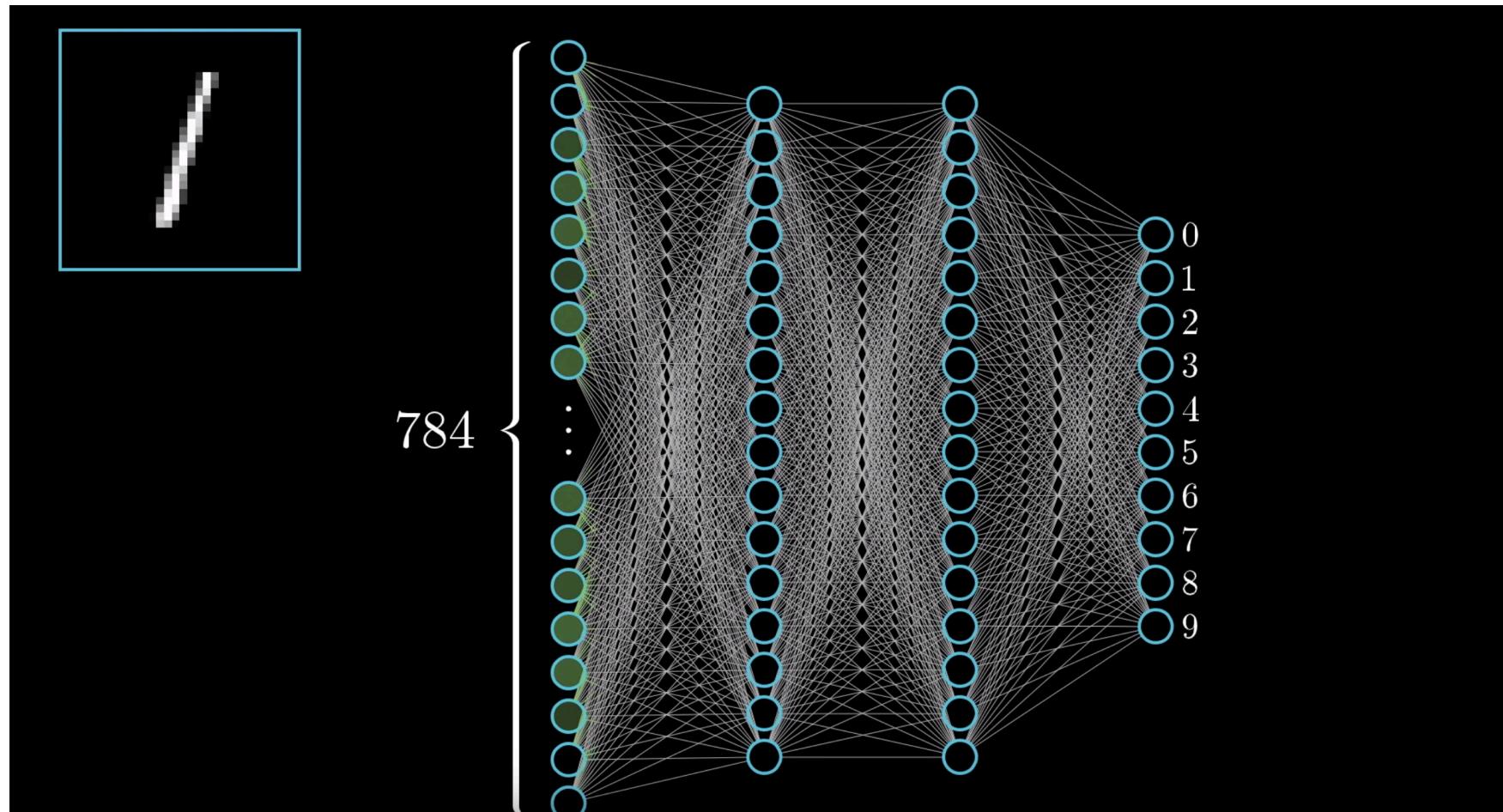
선형변환

비선형 변환

.....

선형변환

Softmax



<https://www.youtube.com/watch?v=aircAruvnKk>

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved. |

$$\text{Likelihood} = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n}$$

공간 데이터 보존: CNN

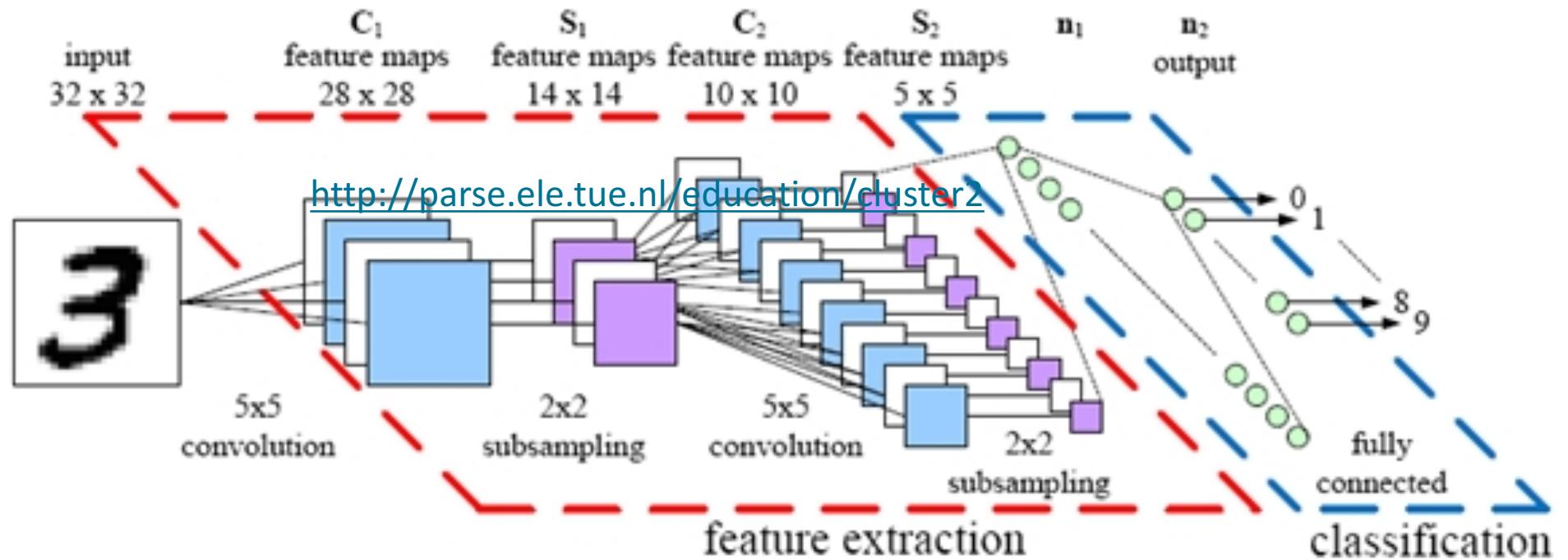
1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

공간 데이터 보존: CNN

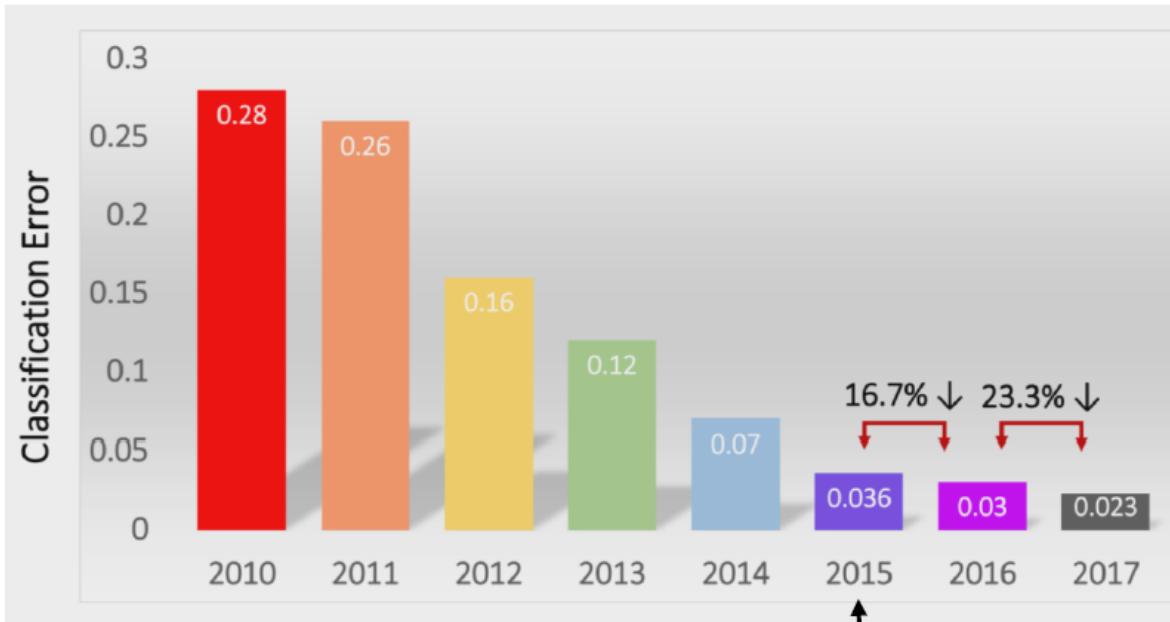


<http://parse.ele.tue.nl/education/cluster2>

Image Data



ImageNet



- AlexNet (2012): First CNN (15.4%)
 - 8 layers
 - 61 million parameters
- ZFNet (2013): 15.4% to 11.2%
 - 8 layers
 - More filters. Denser stride.
- VGGNet (2014): 11.2% to 7.3%
 - Beautifully uniform:
3x3 conv, stride 1, pad 1, 2x2 max pool
 - 16 layers
 - 138 million parameters
- GoogLeNet (2014): 11.2% to 6.7%
 - Inception modules
 - 22 layers
 - 5 million parameters
(throw away fully connected layers)
- ResNet (2015): 6.7% to 3.57%
 - More layers = better performance
 - 152 layers
- CULimage (2016): 3.57% to 2.99%
 - Ensemble of 6 models
- SENet (2017): 2.99% to 2.251%
 - Squeeze and excitation block: network is allowed to adaptively adjust the weighting of each feature map in the convolutional block.

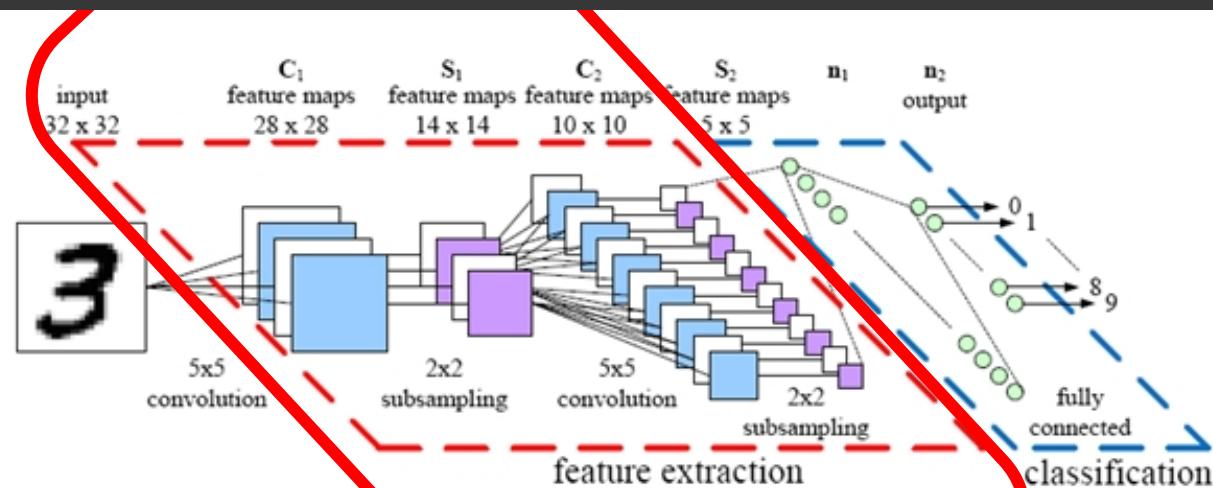
전이학습: Transfer Learning

```
[18] model_ft = models.resnet18(pretrained=True)
    num_ftrs = model_ft.fc.in_features
    model_ft.fc = nn.Linear(num_ftrs, 2)

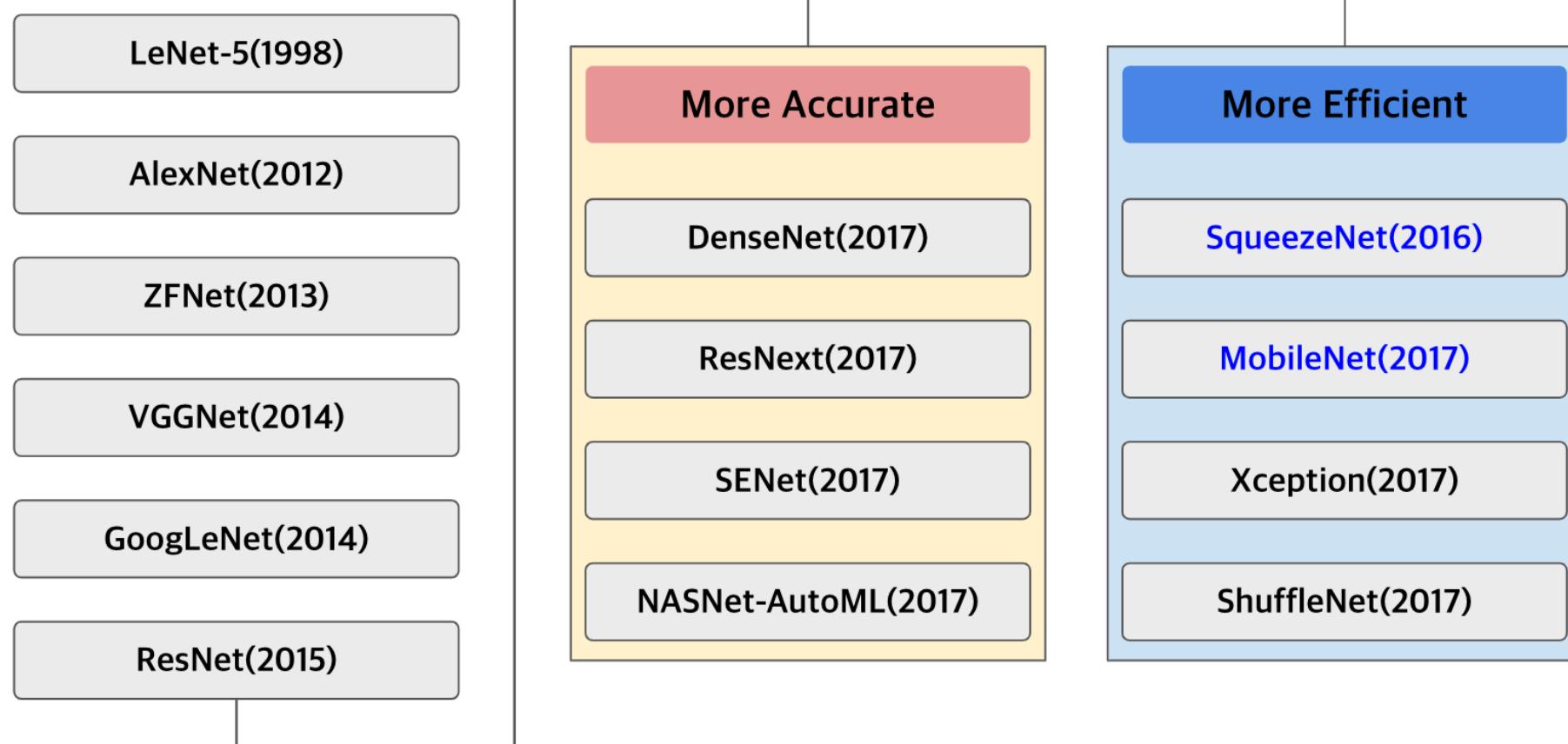
    if torch.cuda.is_available():
        model_ft = model_ft.cuda()
```

↳ Downloading: "<https://download.pytorch.org/models/resnet18-5c106cde.pth>" to .

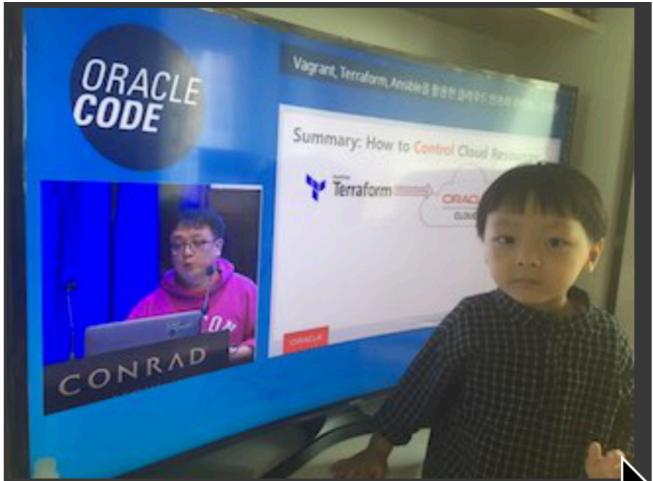
100% |██████████████████| 46827520/46827520 [00:00<00:00, 72851894.93it/s]



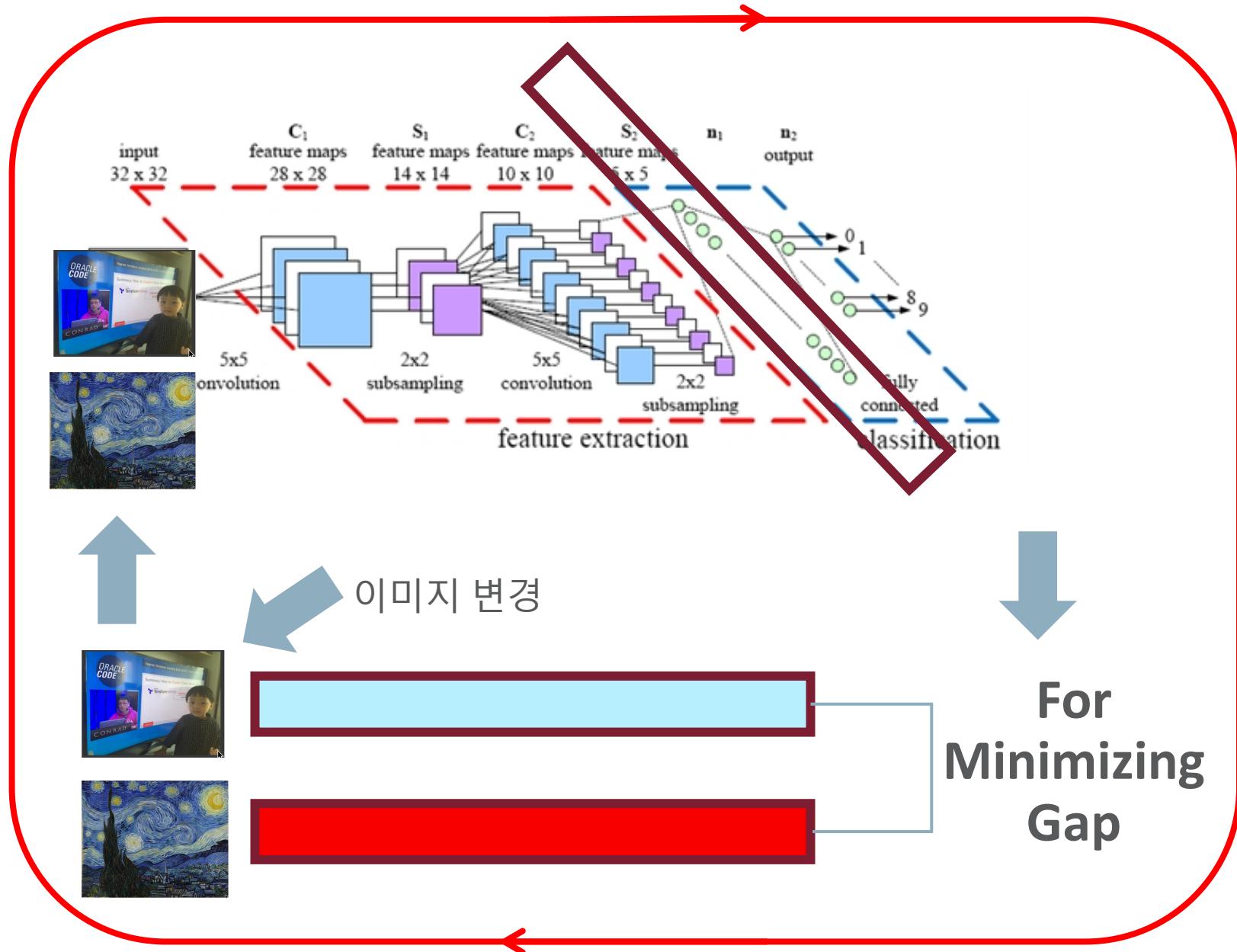
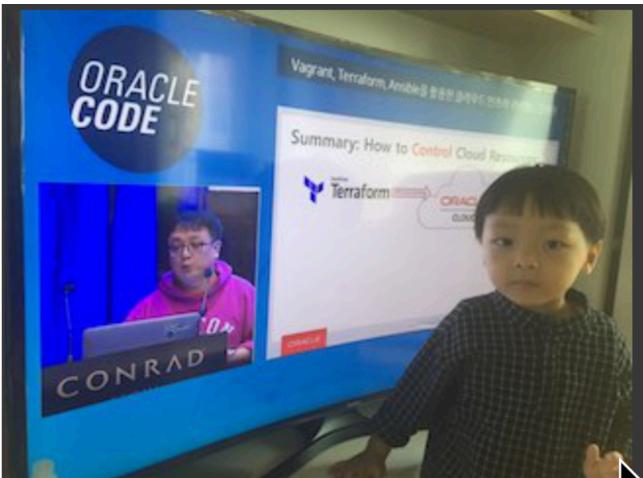
비전 기술 흐름



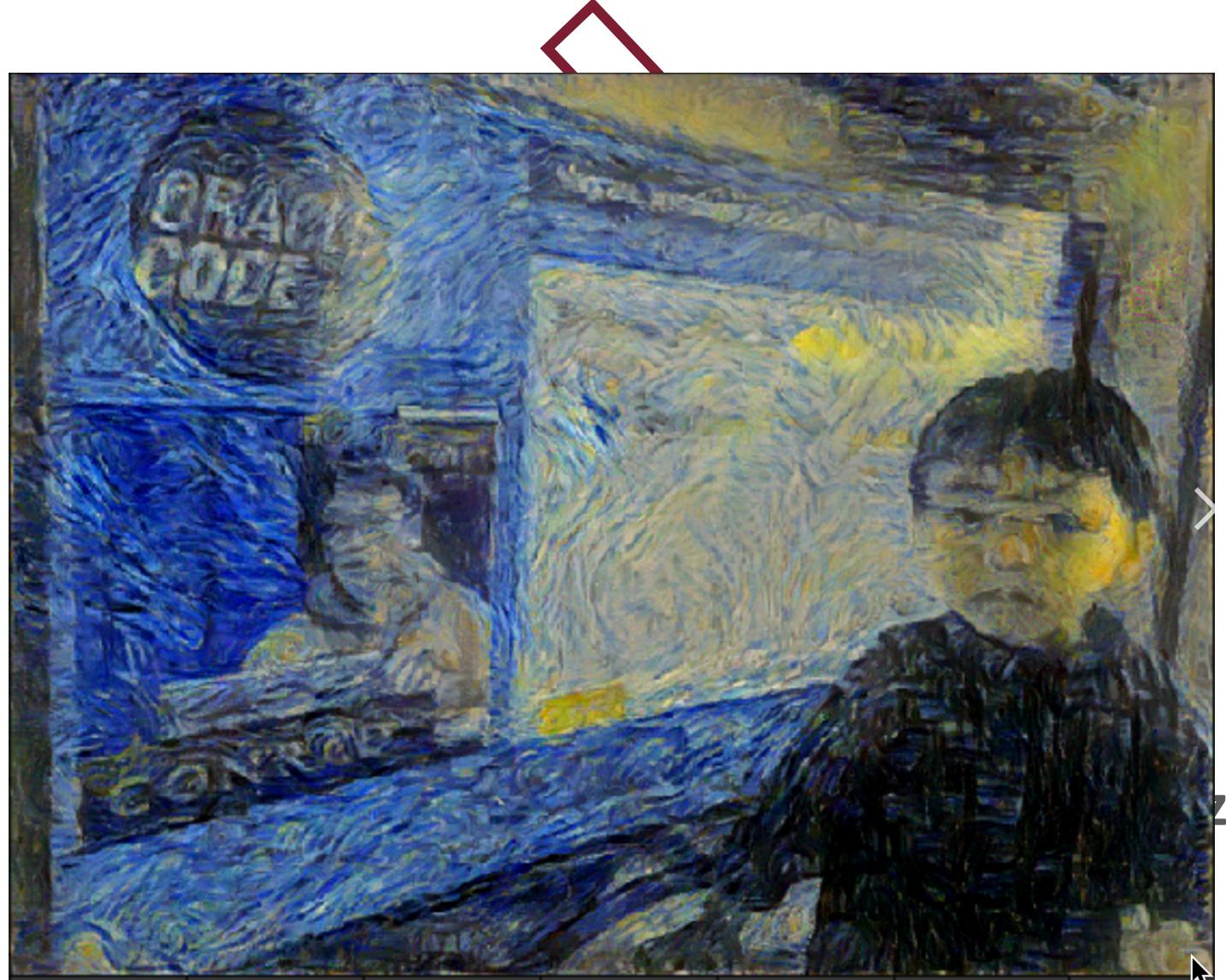
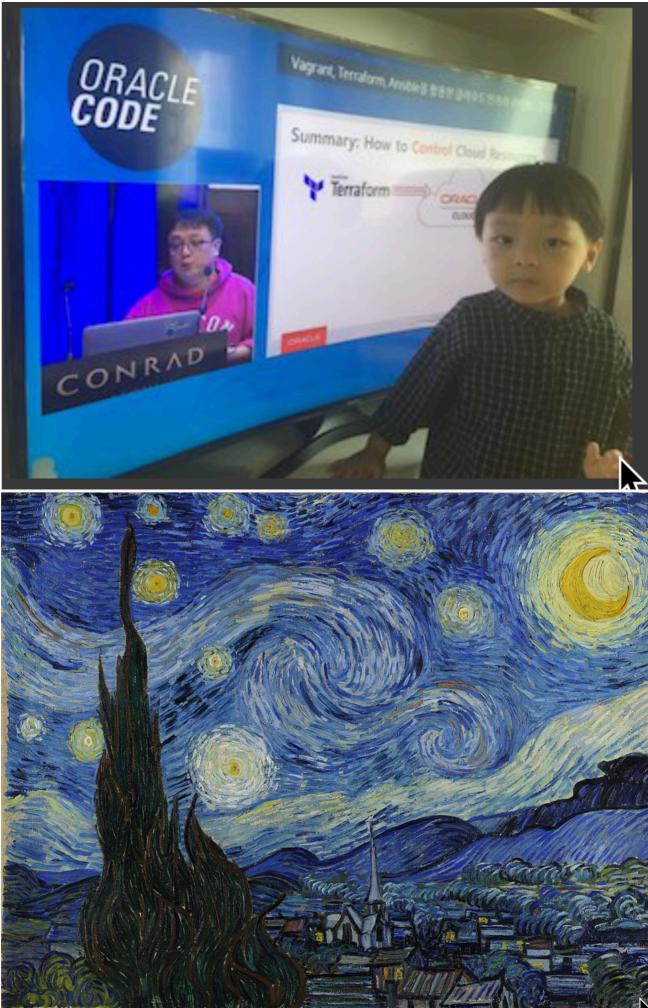
Style Transfer



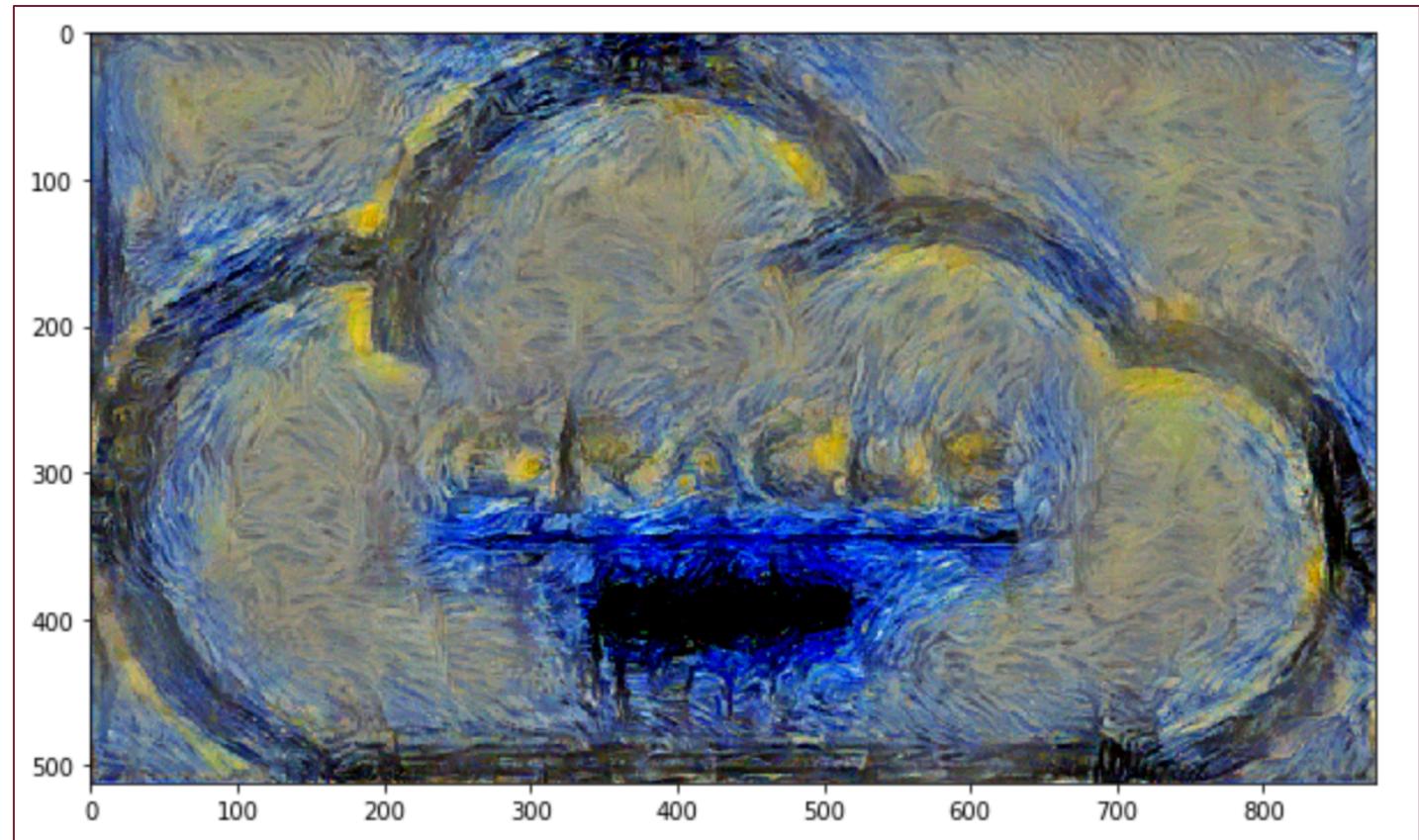
Style Transfer



Style Transfer



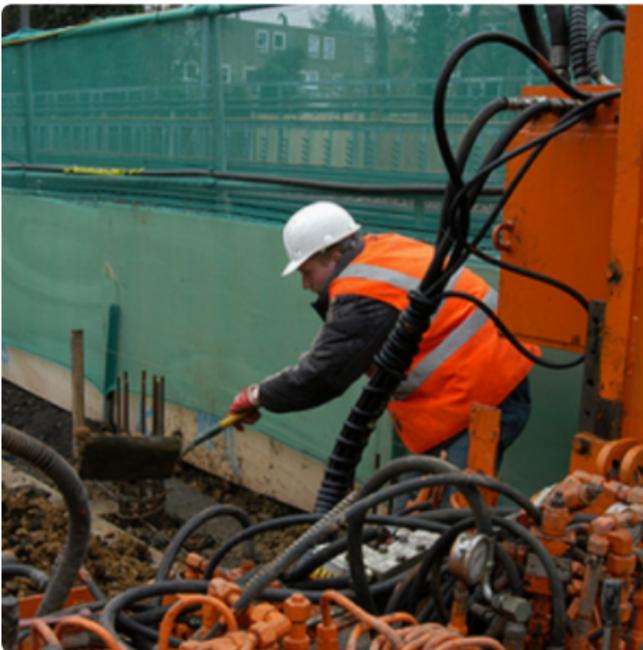
Style Transfer



Object Recognition & Caption



"man in black shirt is playing guitar."

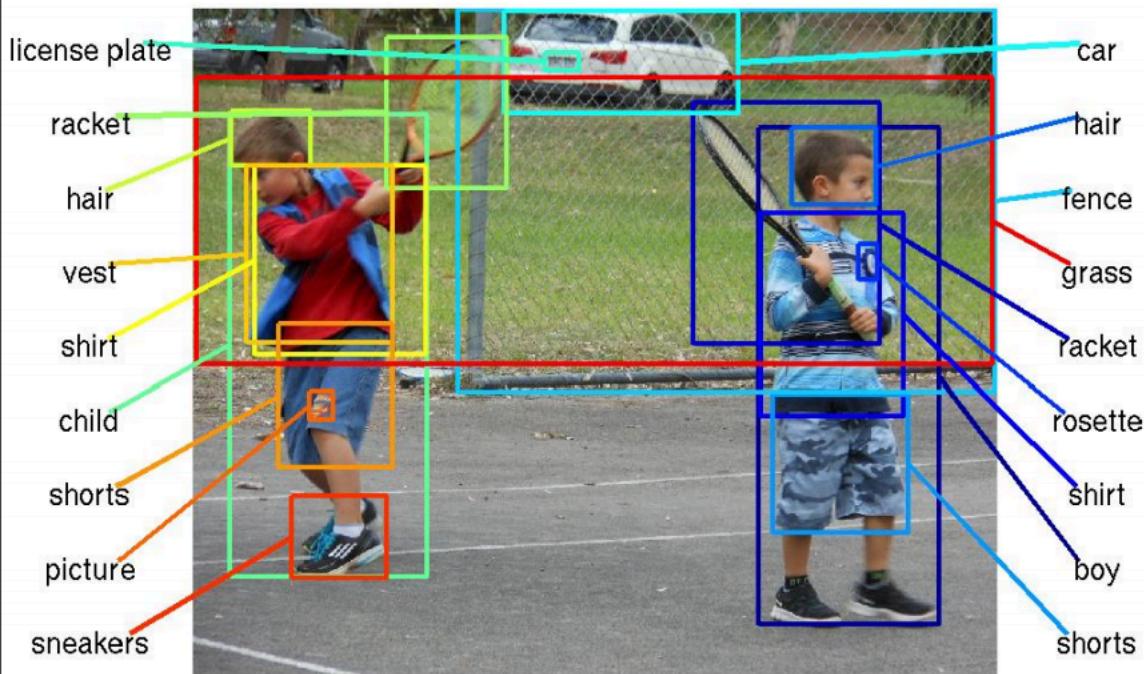


"construction worker in orange safety vest is working on road."

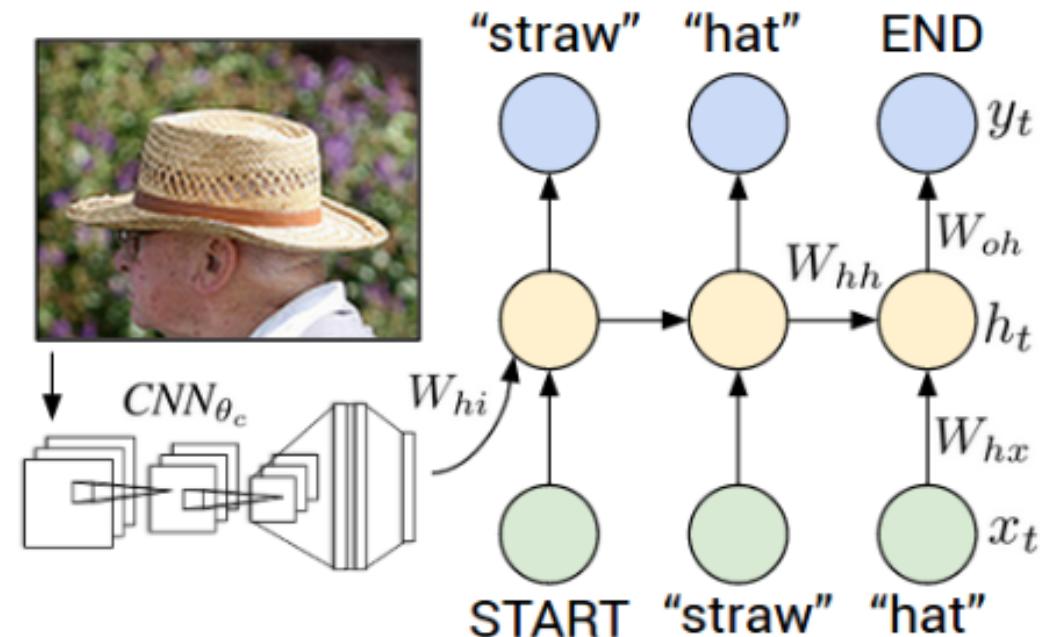
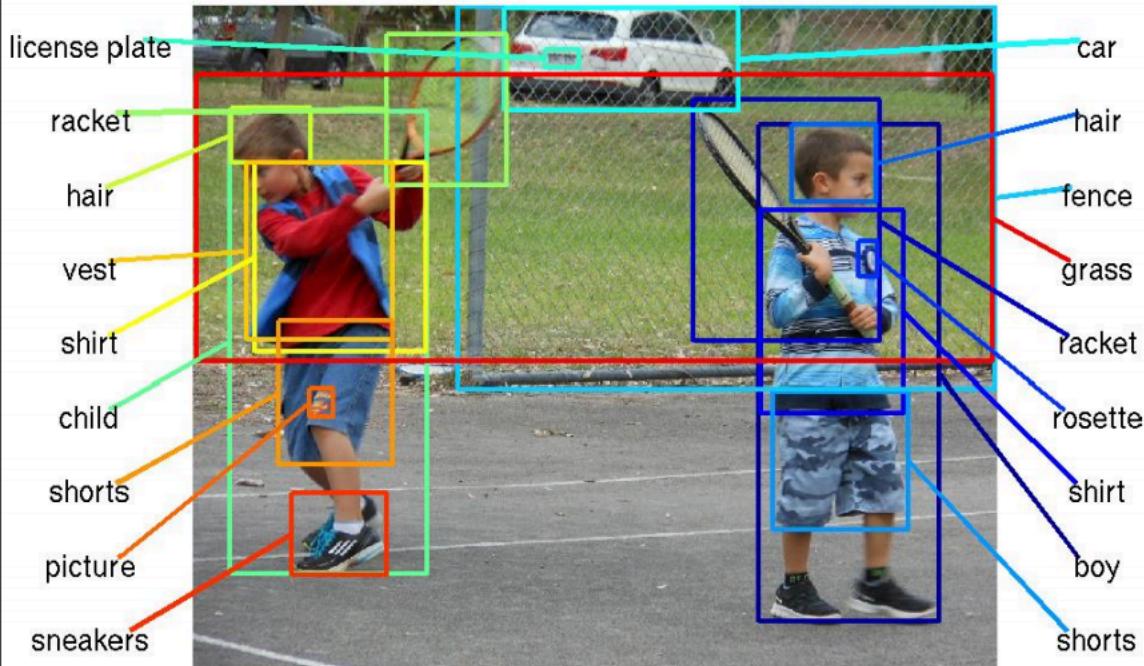


"two young girls are playing with lego toy."

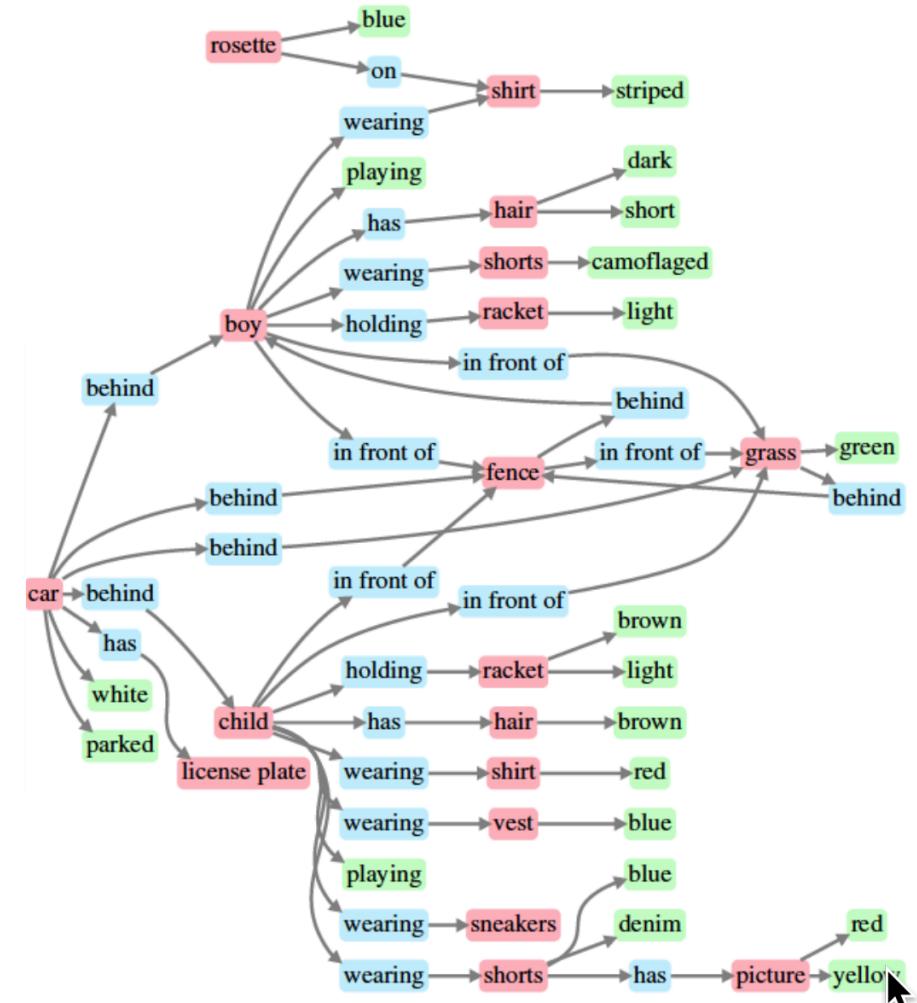
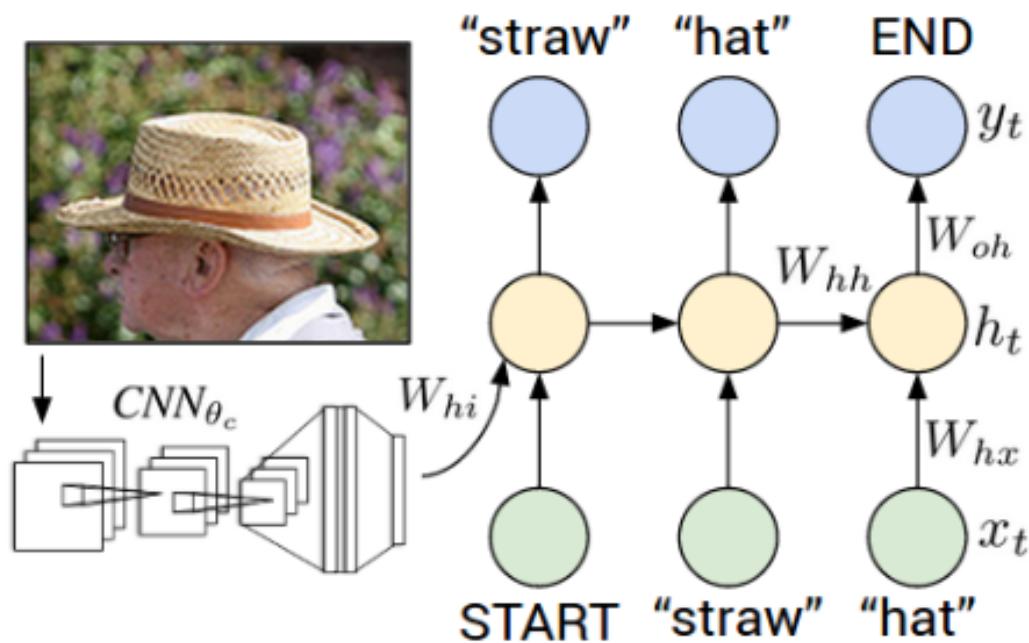
Object Recognition & Caption



Object Recognition & Caption

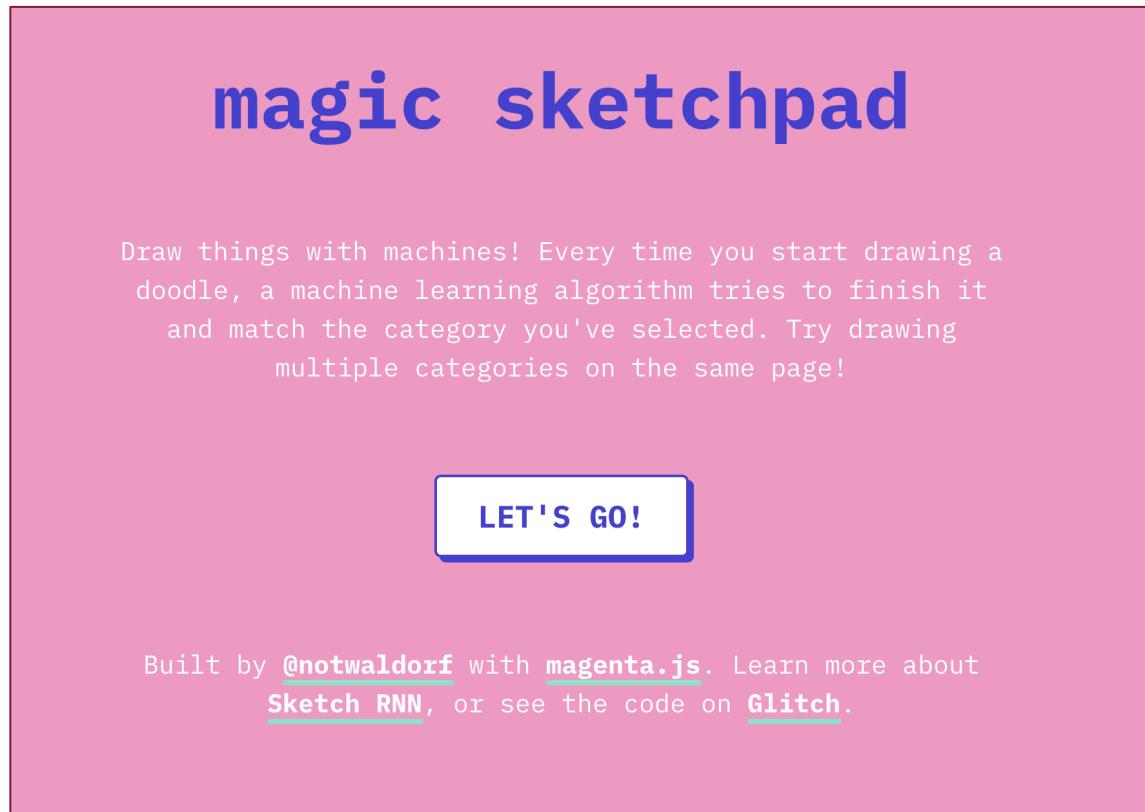


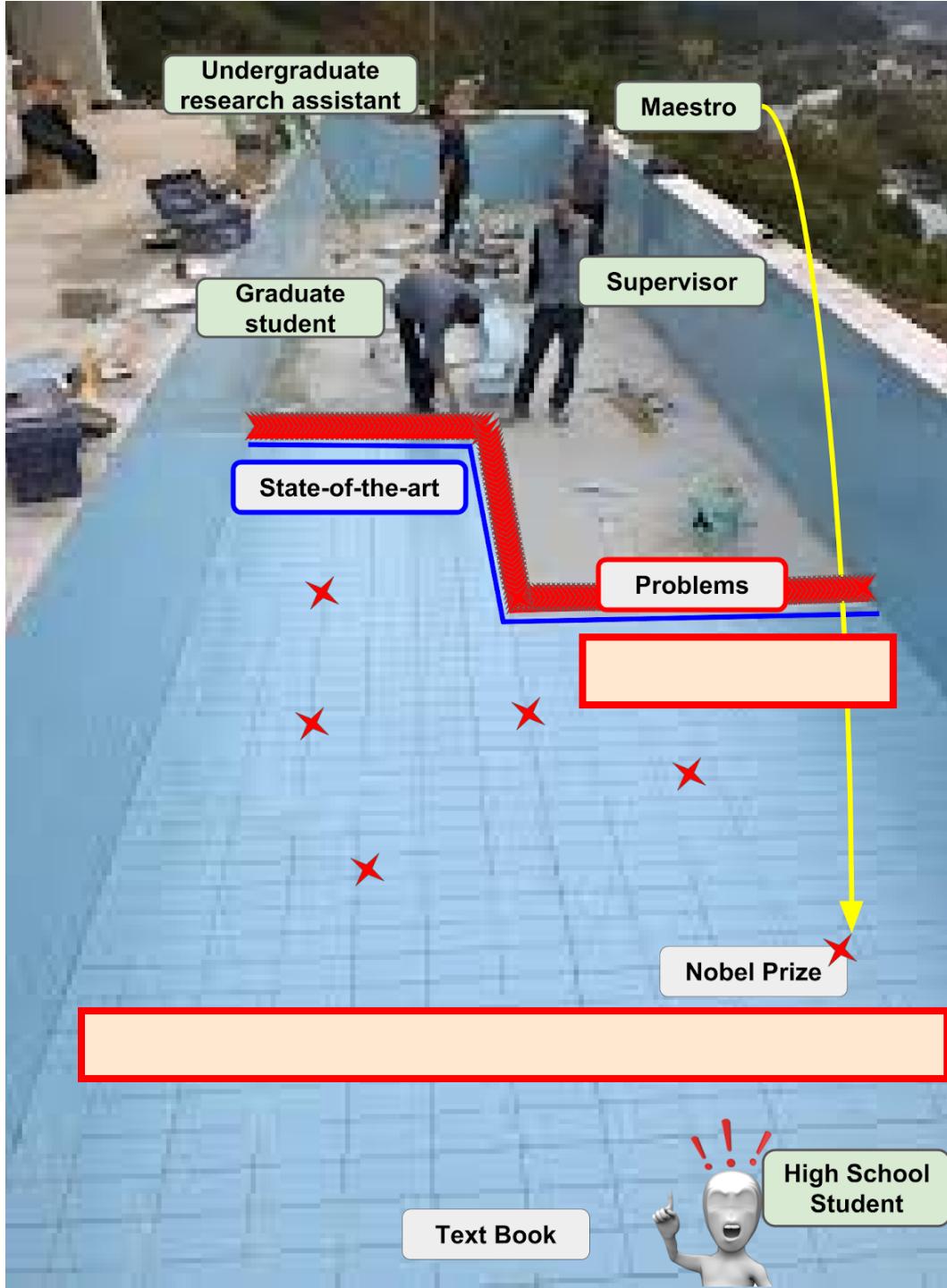
Object Recognition & Caption



Sketch Net

<https://magic-sketchpad.glitch.me/>





머신러닝 어떻게 시작해야 하나요?

입사후 지난 5년 전



빅데이터 플랫폼 구축
제안서

2016. 02. 19



1

ORACLE

ORACLE®

입사후 지난 5년 전



프로젝트 투입: 2016.04 ~ 2016.10

Big Data Project



Just ETL based on Open Technology

입사후 지난 5년: 다음 단계로

The image consists of two main parts. On the left is a presentation slide titled "Data Analysis" in large white letters, set against a background of a man working at a desk and a city skyline. The slide includes the date "2016. 02. 19" and the Oracle logo. On the right is a large, semi-transparent text overlay that reads "Big Data Project" and "Data Analysis" in large, bold, white and grey letters. Below "Data Analysis" is the subtitle "on Open Technology". A red double-equals sign (=) is positioned between "Data Analysis" and "on Open Technology". The overall background of the slide is a light blue gradient.

Big Data Project

Data Analysis

on Open Technology

==

프로젝트 투입: 2016.04 ~ 2016.10

데이터 분석 관련 분야

컴퓨터 공학

수학

통계

Problem
Domain

커리큘럼: 데이터 사이언스 융학 학과

과정명	학점
데이터사이언스기초	3
기초통계	3
빅데이터처리	3
데이터분석언어	3
다면량통계	3
기계학습특론	3
인공지능	3
선형대수응용	3
웹마이닝	3
데이터모델링	3
데이터베이스시스템특론	3
빅데이터플랫폼특론	3
데이터시각화특론	3
자연어처리	3
최적화	3

과정명	학점
응용데이터분석	3
딥러닝	3
서버시스템이해	3
정보보안특론	3
캡스톤프로젝트	3
비즈니스인텔리전스특론	3
빅데이터사례연구 I	3
빅데이터사례연구 II	3
논문연구	3
자료구조/알고리즘	3
IT와지적재산권보호	3
데이터사이언스응용	3
미디어콘텐츠분석기법	3
컴퓨터 비전	3

https://icon.skku.edu/icon/graduate/grad_data_curriculum.do?pager.offset=15

데이터 분석 관련 분야

Top Down



컴퓨터 공학

수학

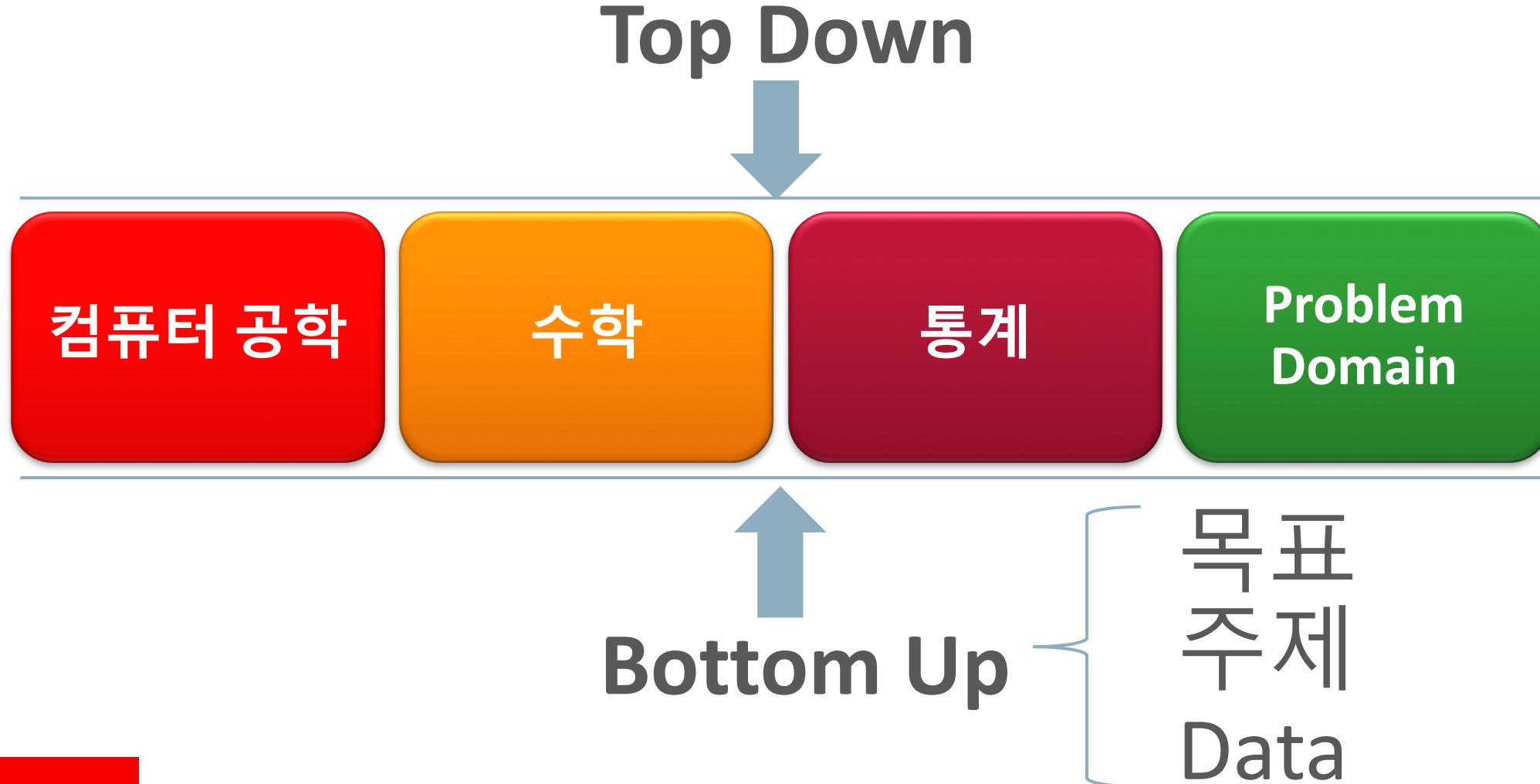
통계

Problem
Domain

입사후 지난 5년: 2017 ~ 현재 시행착오 연속



데이터 분석 관련 분야



Integrated Cloud Applications & Platform Services

ORACLE[®]