

실습 3차

- 사이킷런(Scikit Learn) 실습 -



부산대학교



실습 3차 - 파이썬 필수 라이브러리 설치

I. 머신러닝 모델 구현 과정

1. 주요 머신러닝 용어
2. 머신러닝 파이프라인
3. 머신러닝과 딥러닝 차이점
4. 특성 벡터

II. scikit-learn 실습

1. scikit-learn 개요
2. 주요 데이터 전처리 기법
3. 주피터노트북을 활용한 실습



사전 Python 모듈 설치 여부 확인

■ 실습에 사용될 모듈

- Pandas
- Scikit-learn
- Matplotlib
- Seaborn
- PANDAS-PROFILING

■ 모듈 설치 여부 확인 예시

- 사이킷런(Scikit-learn) 설치여부 확인
 - `pip show pandas`
- 사이킷런이 없을 경우
 - `pip install pandas`

```
PS C:\Users\Hyo EunKang> pip show sklearn
Name: sklearn
Version: 0.0
Summary: A set of python modules for machine learning and data mining
Home-page: https://pypi.python.org/pypi/scikit-learn/
Author: UNKNOWN
Author-email: UNKNOWN
License: None
Location: d:\programdata\python37\lib\site-packages
Requires: scikit-learn
Required-by:
```



머신러닝 모델 구현 과정



1. 주요 머신러닝 용어 (1/2)

용어	정의
라벨(Label)	<ul style="list-style-type: none"> 예측을 하고자 하는 대상 항목 (e.g., 단순 선형 회귀의 y 변수) 예시: 주식의 향후 가격, 사진에 표시된 동물의 종류 등
특성(Feature)	<ul style="list-style-type: none"> 입력 변수 (e.g., 단순 선형 회귀의 x 변수) (분류)스팸 감지 예시 : 이메일 내 단어, 보내는이 주소, 시간, 특정 스팸키워드 등 (회귀)전력 예측 : 과거의 전력 사용량, 기기 정보 등
예(Example)	<ul style="list-style-type: none"> 입력된 데이터들(x)의 특정 객체 x는 벡터라는 것을 나타냄 Label이 있는 예 : Labeled Examples: {features, label} : (x, y) <ul style="list-style-type: none"> Model을 학습시키기 위해 Label이 있는 Examples를 사용 Label이 없는 예 : Unlabeled Examples: {features, label} : ($x, ?$) <ul style="list-style-type: none"> Label이 있는 예를 통해 Label이 없는 Example의 라벨을 예측

housingMedianAge (feature)	totalRooms (feature)	totalBedrooms (feature)	medianHouseValue (label)
15	5612	1283	66900
19	7650	1901	80100
17	720	174	
14	1501	337	73400
20	1454	326	65500

예시 : 캘리포니아 주택 가격 정보가 포함된 데이터세트



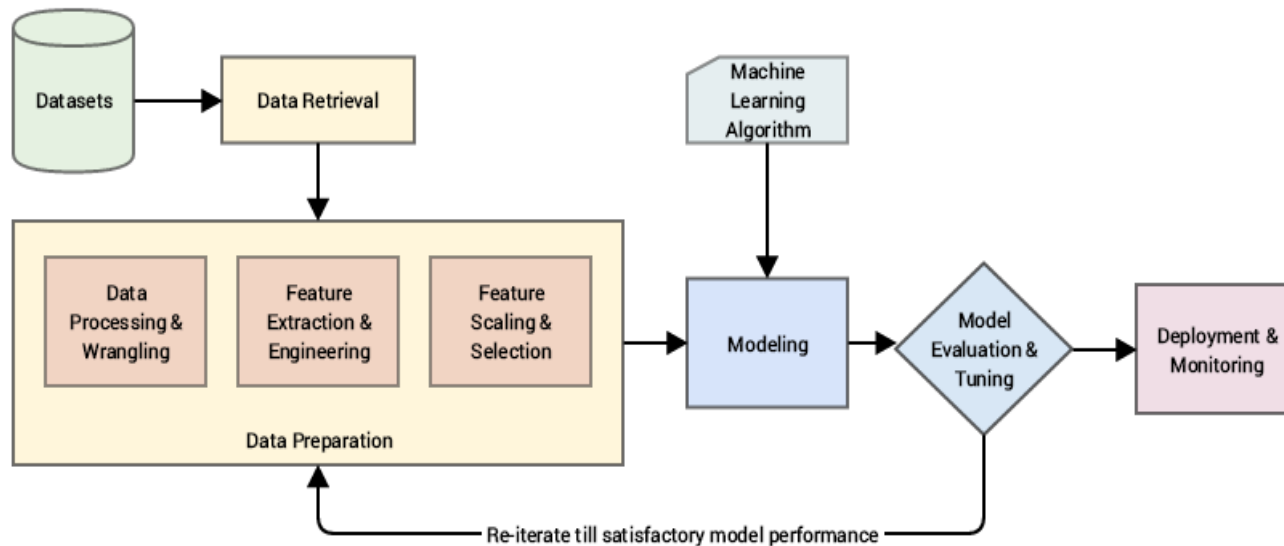
2. 주요 머신러닝 용어 (2/2)

용어	정의
모델(Model)	<ul style="list-style-type: none"> Feature과 Label의 관계를 정의
학습(Training)	<ul style="list-style-type: none"> 모델이 Feature과 Label의 관계를 점차적으로 학습하는 것
추론(Inference)	<ul style="list-style-type: none"> 학습된 모델을 Label이 없는 예(Example)에 적용하는 것
회귀 모델 (Regression Model)	<ul style="list-style-type: none"> 연속적인 값을 예측하는데 사용 예시 : 유저가 유튜브 광고를 클릭할 확률
분류 모델 (Classification Model)	<ul style="list-style-type: none"> 불연속적인 값을 예측하는데 사용 예시 : 이메일 스팸 여부



3. 머신러닝 파이프라인

■ 머신러닝 모델 구축을 위한 파이프라인



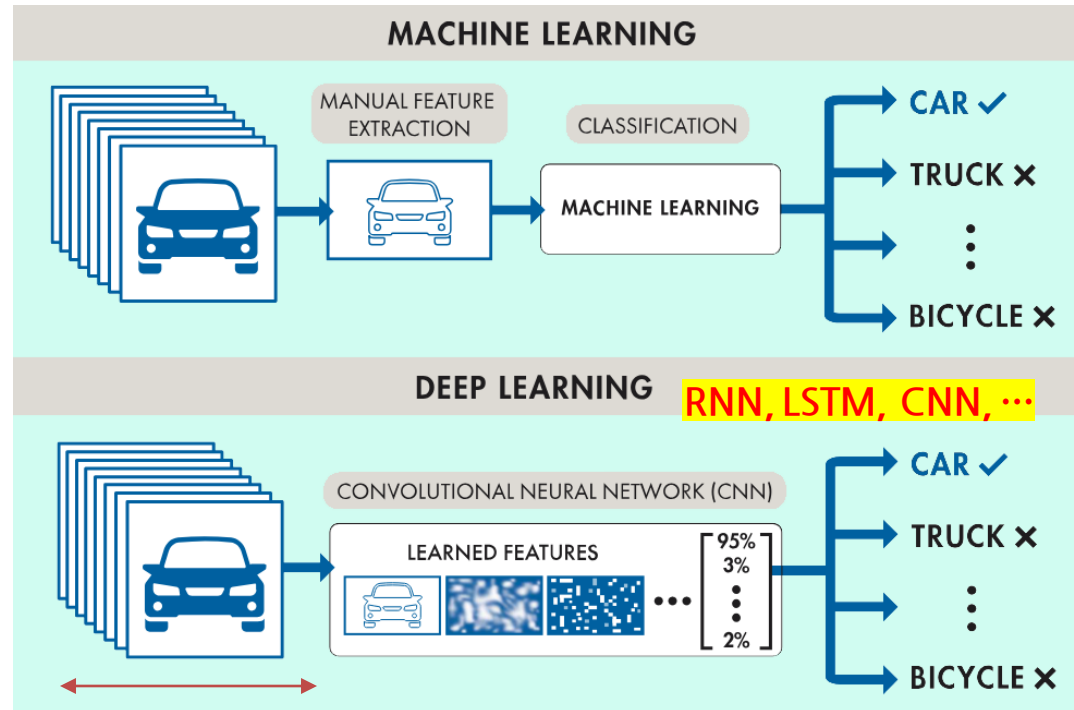
A standard machine learning pipeline (source: Practical Machine Learning with Python, Apress/Springer)

표현	정의
Feature Extraction	차원축소 등 새로운 중요 피처를 추출
Feature Selection	기존 피처에서 원하는 피처만 (변경하지 않고) 선택하는 과정



4. 머신러닝과 딥러닝 차이점

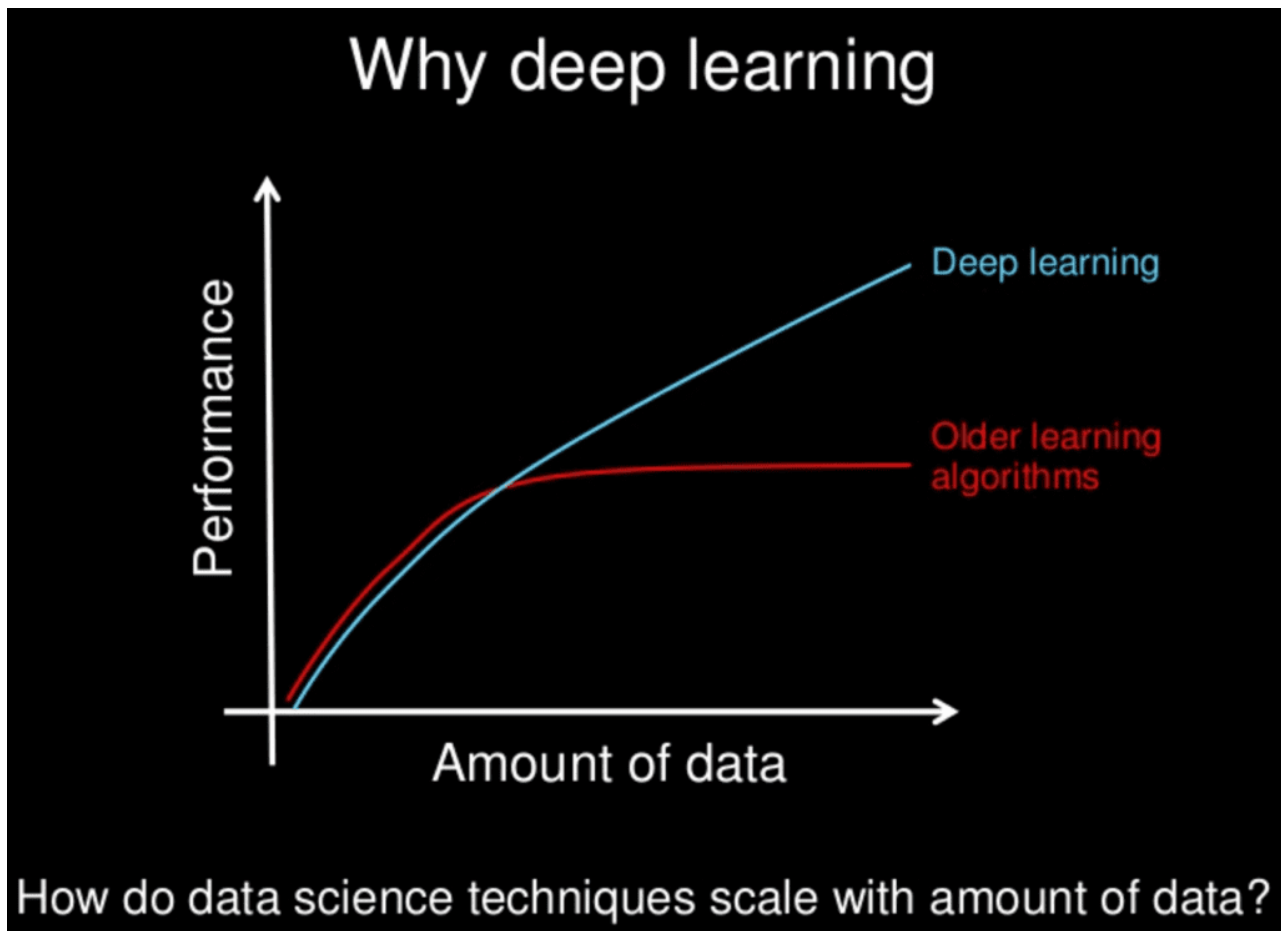
■ 머신러닝과 딥러닝 차이점



객체 인식에 사용되는 머신러닝과 딥러닝 기술(자료:매스웍스)



4. 머신러닝과 딥러닝 차이점



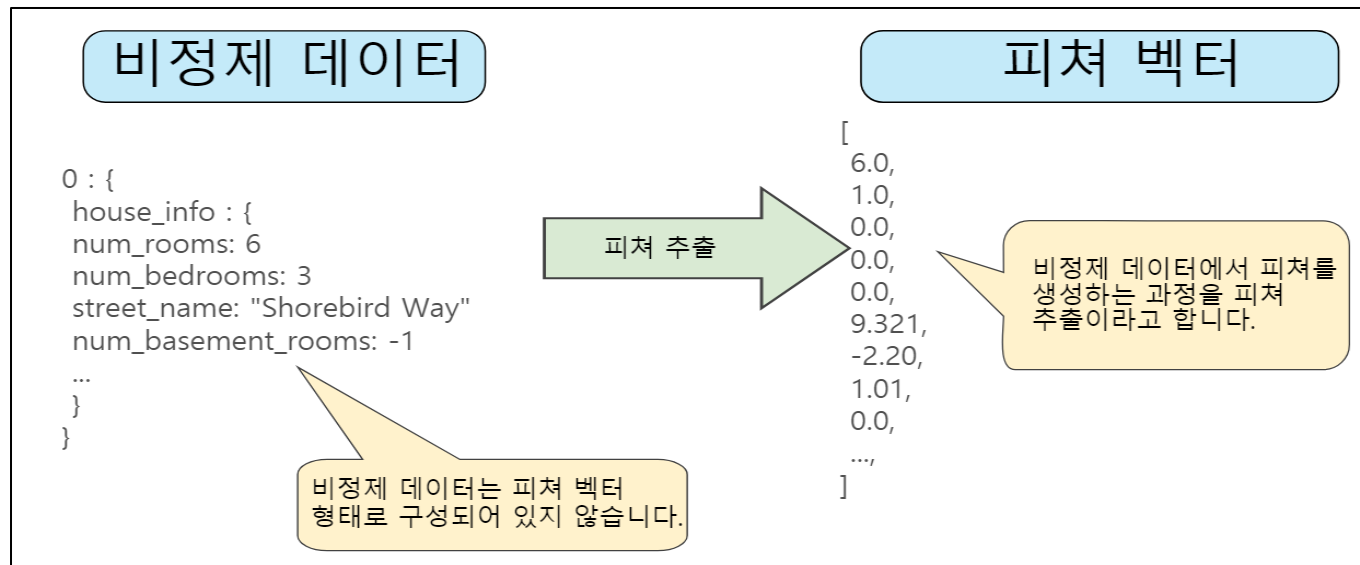
(자료: Why Deep Learning? (Andrew Ng))



5. 특성 벡터 (1/3)

■ 특성 벡터 (Feature Vector)

- 기존의 프로그래밍에서는 코드에 중점을 두었으나, 머신러닝 모델 구현은 특성(Feature)의 표현에 중점을 둠
- 개발자는 특성을 추가하고 개선하여 모델을 다듬어 나감



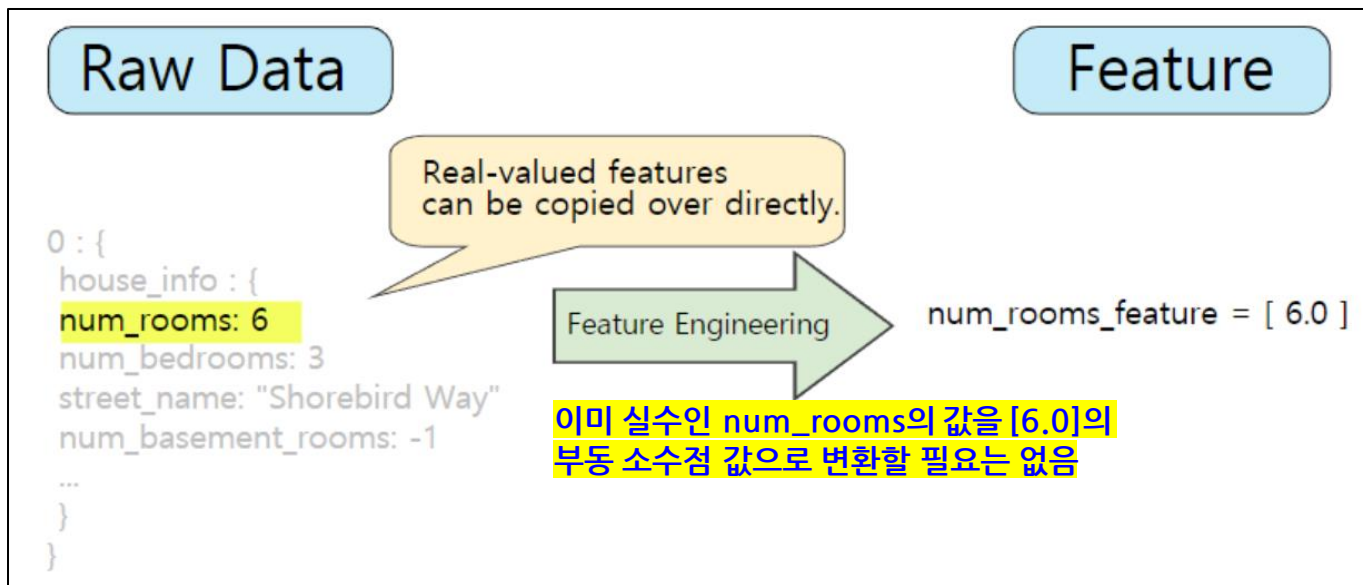
Feature Engineering: 원시 데이터(Raw Data)를 실수 벡터로 표현된 특성 벡터(Feature Vector)로 만드는 과정
<https://developers.google.com/machine-learning/crash-course/representation/feature-engineering>



5. 특성 벡터 (1/3)

■ 특성 벡터 (Feature Vector)

- 기존의 프로그래밍에서는 코드에 중점을 두었으나, 머신러닝 모델 구현은 특성(Feature)의 표현에 중점을 둬
- 개발자는 특성을 추가하고 개선하여 모델을 다듬어 나감



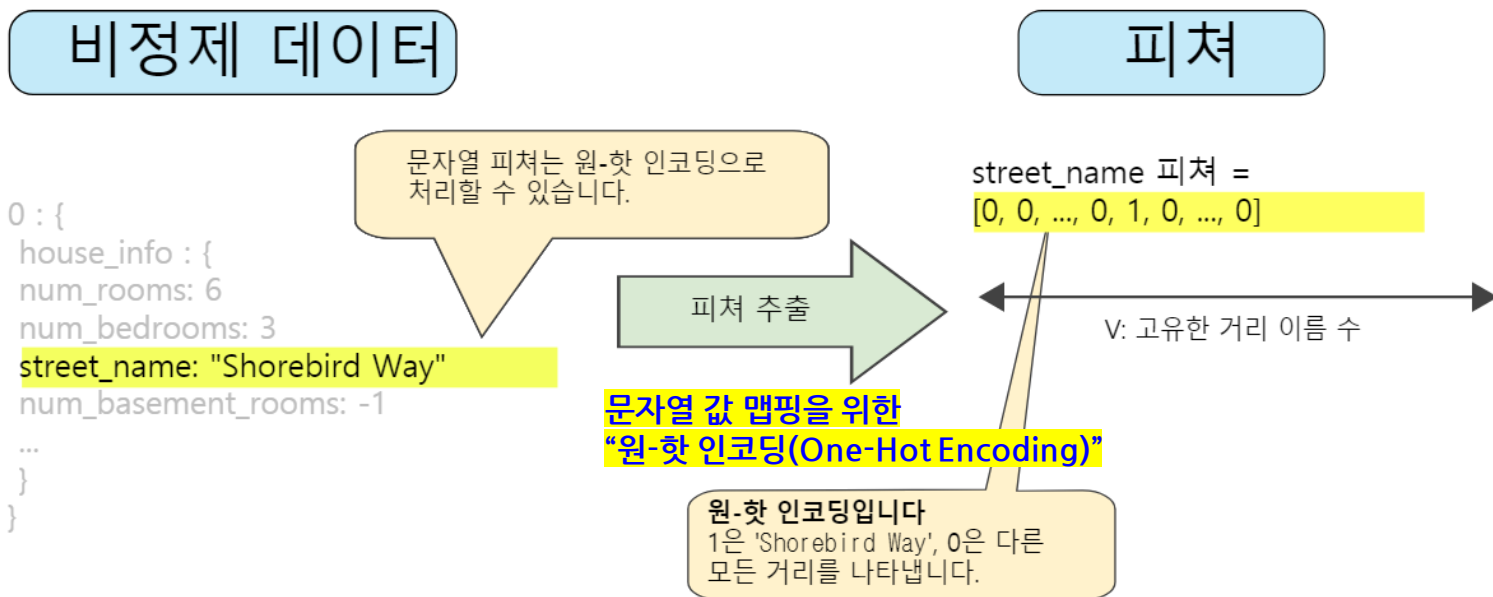
Feature Engineering: 원시 데이터(Raw Data)를 실수 벡터로 표현된 특성 벡터(Feature Vector)로 만드는 과정
<https://developers.google.com/machine-learning/crash-course/representation/feature-engineering>



5. 특성 벡터 (1/3)

■ 특성 벡터 (Feature Vector)

- 기존의 프로그래밍에서는 코드에 중점을 두었으나, 머신러닝 모델 구현은 특성(Feature)의 표현에 중점을 둠
- 개발자는 특성을 추가하고 개선하여 모델을 다듬어 나감



Feature Engineering: 원시 데이터(Raw Data)를 실수 벡터로 표현된 특성 벡터(Feature Vector)로 만드는 과정
<https://developers.google.com/machine-learning/crash-course/representation/feature-engineering>



II. scikit-learn 실습



1. scikit-learn(사이킷런) 개요

■ 파이썬 머신러닝 라이브러리

- 파이썬 라이브러리인 NumPy와 SciPy를 기반으로 만들어짐

scikit-learn

Machine Learning in Python

Getting Started Release Highlights for 0.23 GitHub

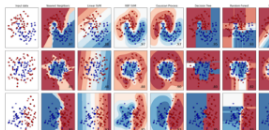
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification 분류

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



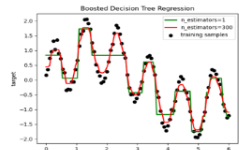
Examples

Regression 회귀

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...




Examples

Clustering 군집화

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



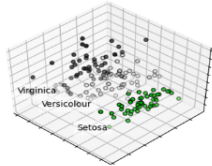
Examples

Dimensionality reduction 차원 축소

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



Examples

Model selection 모델 선택

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning

Algorithms: grid search, cross validation, metrics, and more...



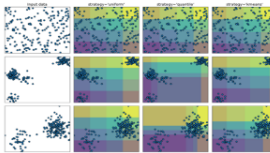
Examples

Preprocessing 전처리

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.

Algorithms: preprocessing, feature extraction, and more...



Examples

지도학습 프레임워크

비지도학습 프레임워크

비지도학습 프레임워크

<https://scikit-learn.org/stable/>



■ 사이킷런의 주요 API/모듈

모듈타입	모듈명	모듈설명
예제 데이터	sklearn.dataset	사이킷런에 내장되어 있는 예제로 제공하는 데이터세트
특성(Feature) 처리	sklearn.preprocessing	데이터 전처리에 필요한 다양한 가공 기능 제공 (정규화, 스케일링 등)
	sklearn.feature_selection	알고리즘에 큰 영향을 미치는 feature를 우선순위대로 selection하는 기능 제공
	sklearn.feature_extraction	데이터에서 벡터화 된 feature를 추출하는 데 사용
	sklearn.decomposition	차원 축소 관련 알고리즘 제공 (PCA, NMF 등)
데이터 분리, 검증 & 파라미터 튜닝	sklearn.model_selection	교차 검증을 위한 학습용/테스트용 분리 등의 API 제공
평가	sklearn.metrics	머신러닝 모델에 대한 다양한 성능 측정 방법 제공 (정확도, 정밀도, 재현율, ROC-AUC, RMSE 등)



■ 사이킷런의 주요 API/모듈

모듈타입	모듈명	모듈설명
머신러닝 알고리즘	sklearn.ensemble	앙상블 알고리즘 제공 (랜덤 포레스트 , 에이다 부스트, 그래디언트 부스팅 등)
	sklearn.linear_model	선형 회귀, 릿지(Ridge), 라쏘(Lasso) 및 로지스틱 회귀 등 회귀 관련 알고리즘 제공
	sklearn.naive_bayes	나이브 베이즈 알고리즘 제공 (가우시안 NB, 다항분포 NB 등)
	sklearn.neighbors	최근접 이웃 알고리즘 제공 (k-NN 등)
	sklearn.svm	서포트 벡터 머신 알고리즘 제공
	sklearn.tree	의사결정트리 알고리즘 제공
	sklearn.cluster	비지도학습 클러스터링 알고리즘 제공 (k-means, DBSCAN 등)



2. 주요 데이터 전처리 기법

■ 사이킷런에서 주로 사용되는 데이터 전처리 사항

- 모델의 성능을 측정하려면, 이전에 사용하지 않은 새로운 데이터(테스트 세트)를 모델에 적용해야 함
 - 학습 세트(training set) : 머신러닝 모델을 만들 때 사용
 - 테스트 세트(test set) : 모델이 얼마나 잘 작동하는지 측정/평가하는 데 사용
- 머신러닝 알고리즘에서의 입력 데이터
 - 결측값(NaN, Null)을 허용하지 않음 → Null 값은 다른 값으로 변환해야 함
 - 문자열 값을 입력값으로 허용하지 않음 → 문자열 값을 인코딩해서 숫자형으로 변환해야 함
- 데이터 스케일링과 정규화
 - 데이터 스케일링(Feature Scaling) : 서로 다른 변수의 값 범위를 일정한 수준으로 맞추는 작업(표준화, 정규화 등)
 - 표준화(Standardization) : 데이터 피처의 각각이 평균이 0이고 분산이 1인 가우시안 정규분포를 가진 값으로 변환



[실습] Boston 집값 예측

```
[1]: # 사용할 라이브러리 가져오기
import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import seaborn as sns

%matplotlib inline
```

```
[2]: # 보스턴 집값 데이터 가져오기
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression

boston_dataset = load_boston()
```

```
[3]: boston_dataset.keys()
```

```
[3]: dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
```

```
[4]: # 보스턴 데이터셋 설명 출력
print(boston_dataset.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
-----

**Data Set Characteristics:**

    :Number of Instances: 506
```



[실습] Boston 집값 예측

```
[5]: # 보스턴 데이터셋의 Feature 목록 확인
print(boston_dataset.feature_names)

['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

데이터 속성 (Feature) 및 설명

- CRIM : 범죄율
- INDUS : 비소매상업지역 면적 비율
- NOX : 일산화질소 농도
- RM : 주택당 방 수
- LSTAT : 인구 중 하위 계층 비율
- B : 인구 중 흑인 비율
- PTRATIO : 학생/교사 비율
- ZN : 25,000 평방피트를 초과 거주지역 비율
- CHAS : 찰스강의 경계에 위치 유무 (1은 위치, 0은 위치하지 않음)
- AGE : 1940년 이전에 건축된 주택의 비율
- RAD : 방사형 고속도로까지의 거리
- DIS : 직업센터의 거리
- TAX : 재산세율
- MEDV : 본인 소유의 주택가격(중앙값) (단위:\$1,000)

```
[6]: # 보스턴 데이터셋을 DataFrame 형식으로 변환
boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
boston.head()
```

```
[6]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33



```
[7]: # Target variable "MEDV" 가 없기 때문에 생성해주어야 함
      boston['MEDV'] = boston_dataset.target
```

```
[8]: boston.head()
```

```
[8]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

데이터 전처리

```
[9]: # NULL 값이 포함된 Feature 확인
      boston.isnull().sum()
```

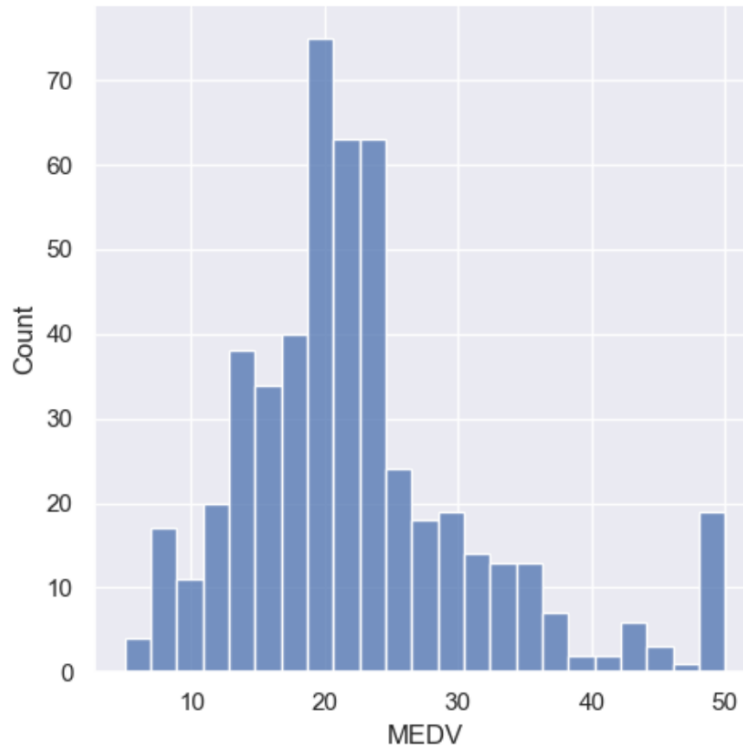
```
[9]: CRIM      0
      ZN      0
      INDUS  0
      CHAS   0
      NOX    0
      RM     0
      AGE    0
      DIS    0
      RAD    0
      TAX    0
      PTRATIO 0
      B      0
      LSTAT  0
      MEDV   0
      dtype: int64
```



데이터 탐색 🔍

MEDV(주택 가격) 분포

```
[10]: sns.set(rc={'figure.figsize':(11.7,8.27)})  
      sns.displot(boston['MEDV'])  
      plt.show()
```

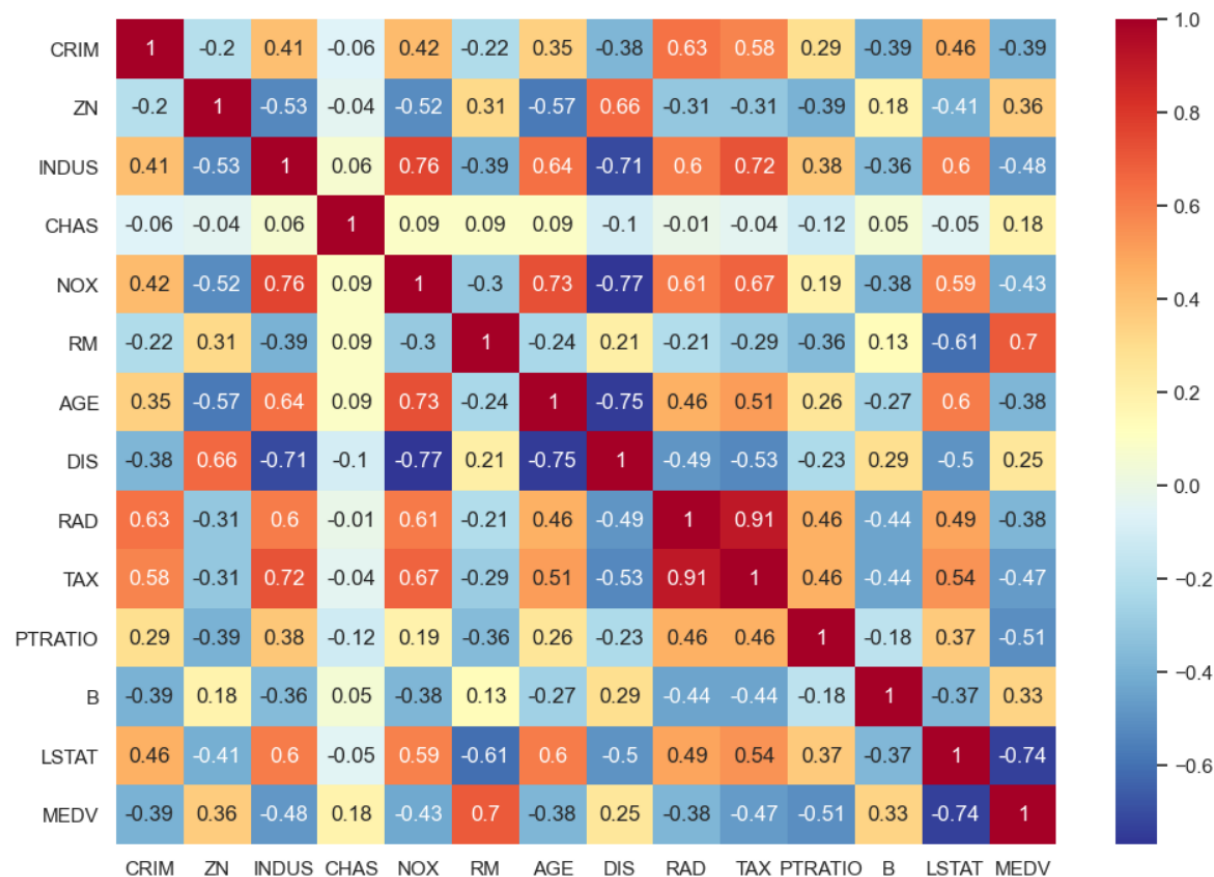


Visualization & Observation

Regression model을 만들기 위해 Target 변수"MEDV"와 높은 상관관계를 갖는 Feature를 선택함

```
[11]: correlation_matrix = boston.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True, cmap='RdYlBu_r',)
```

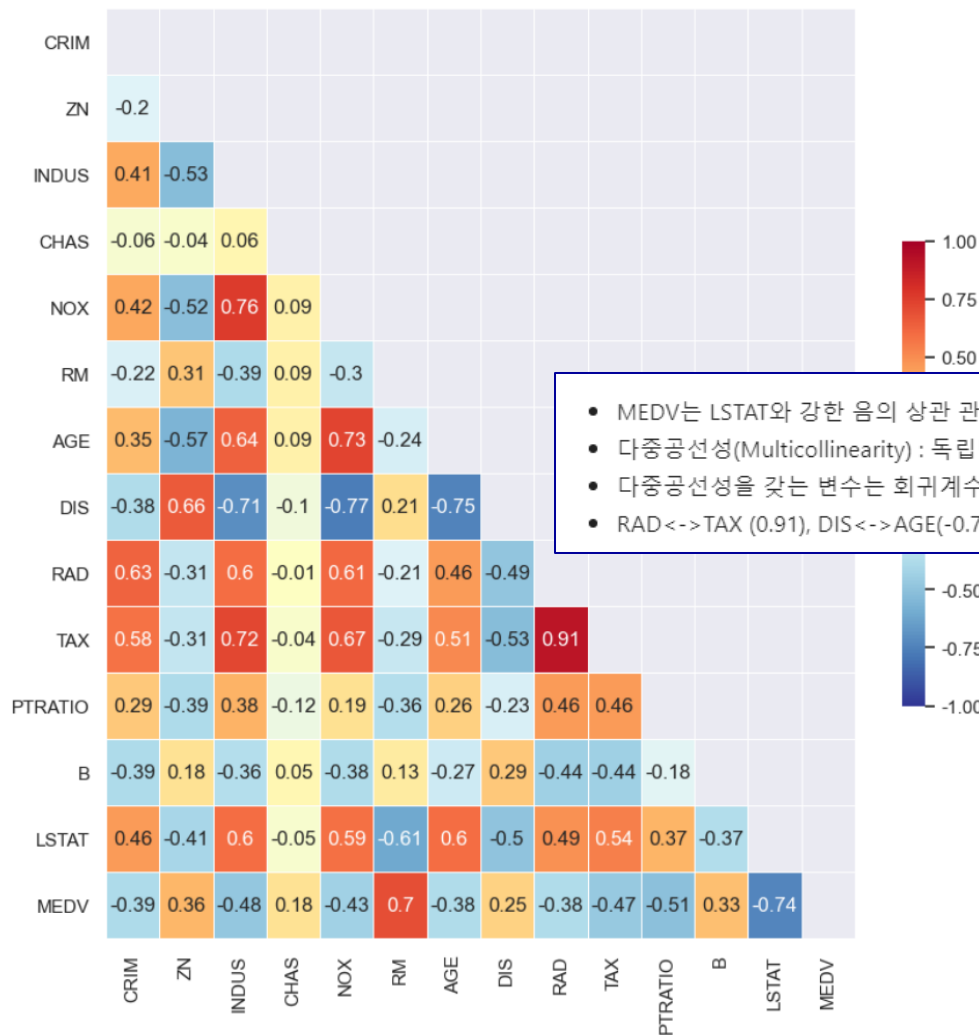
[11]: <AxesSubplot:>



[실습] Boston 집값 예측

```
: mask = np.zeros_like(correlation_matrix, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

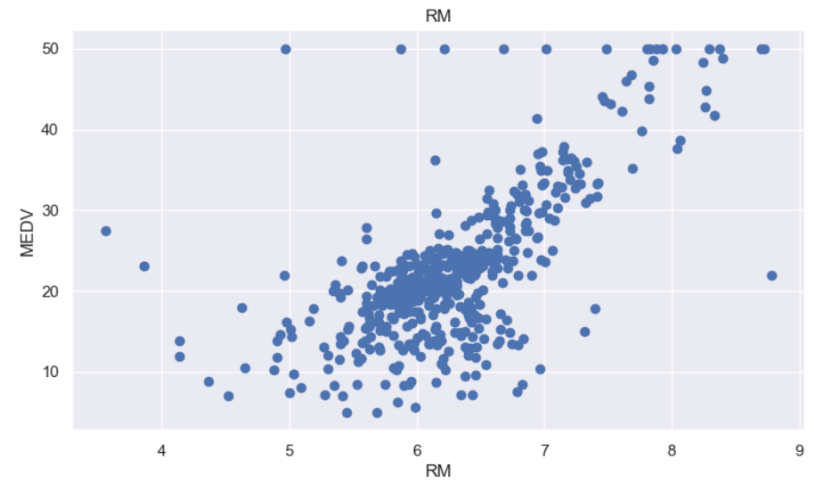
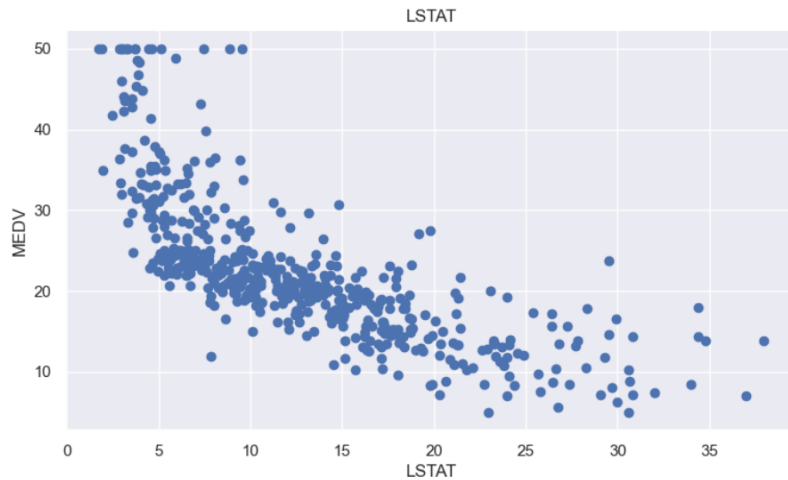
plt.figure(figsize=(10, 10))
sns.heatmap(correlation_matrix, annot=True, mask=mask, cmap='RdYlBu_r', linewidths=.5, cbar_kws={"shrink": .5}, vmin=-1, vmax=1)
plt.show()
```



- MEDV는 LSTAT와 강한 음의 상관 관계가 있고, RM과 강한 양의 상관 관계가 있음
- 다중공선성(Multicollinearity) : 독립변수들이 강한 선형관계를 가지고 있는 형태
- 다중공선성을 갖는 변수는 회귀계수의 표준오차를 증가시키고, 회귀 계수를 불안정하게 함
- RAD <-> TAX (0.91), DIS <-> AGE (-0.75)



```
plt.figure(figsize=(20, 5))  
features = ['LSTAT', 'RM']  
target = boston['MEDV']  
  
for i, col in enumerate(features):  
    plt.subplot(1, len(features), i+1)  
    x = boston[col]  
    y = target  
    plt.scatter(x, y, marker='o')  
    plt.title(col)  
    plt.xlabel(col)  
    plt.ylabel('MEDV')
```



데이터 시각화 리포트 툴 ¶

```
import pandas_profiling
import matplotlib
from matplotlib import font_manager, rc

matplotlib.rcParams['axes.unicode_minus'] = False
#그래프에서 마이너스 기호가 표시되도록 하는 설정입니다.

# 레포트 생성 --> html 파일로 저장하기
report = boston.profile_report()
report.to_file('boston_report.html')
```

Summarize dataset: 100%  27/27 [00:25<00:00, 1.51s/it, Completed]

Generate report structure: 100%  1/1 [00:03<00:00, 3.74s/it]

Render HTML: 100%  1/1 [00:04<00:00, 4.15s/it]

Export report to file: 100%  1/1 [00:00<00:00, 14.49it/s]



Overview

Overview
Warnings 48
Reproduction

Dataset statistics

Number of variables	14
Number of observations	506
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	55.5 KiB
Average record size in memory	112.3 B

Variable types

Numeric	13
Categorical	1

Variables

CRIM

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	504
Distinct (%)	99.6%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	3.613523557

Minimum	0.00632
Maximum	88.9762
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	4.1 KiB

Toggle details



```
: X = pd.DataFrame(np.c_[boston['LSTAT'], boston['RM']], columns = ['LSTAT', 'RM'])
Y = boston['MEDV']

RANDOM_SEED = 5
SPLIT_SIZE = 0.2

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = SPLIT_SIZE, random_state=RANDOM_SEED)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)

(404, 2)
(102, 2)
(404,)
(102,)
```

모델 학습

```
: from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)

: LinearRegression()
```



모델 검증

```
# model evaluation for training set
y_train_predict = lin_model.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict)))
r2 = r2_score(Y_train, y_train_predict)

print("The model performance for training set")
print("-----")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("\n")

# model evaluation for testing set
y_test_predict = lin_model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
r2 = r2_score(Y_test, y_test_predict)

print("The model performance for testing set")
print("-----")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
The model performance for training set
-----
RMSE is 5.637129335071195
R2 score is 0.6300745149331701
```

```
The model performance for testing set
-----
RMSE is 5.137400784702911
R2 score is 0.6628996975186952
```

```
lin_model.coef_
```

```
array([-0.71722954,  4.58938833])
```

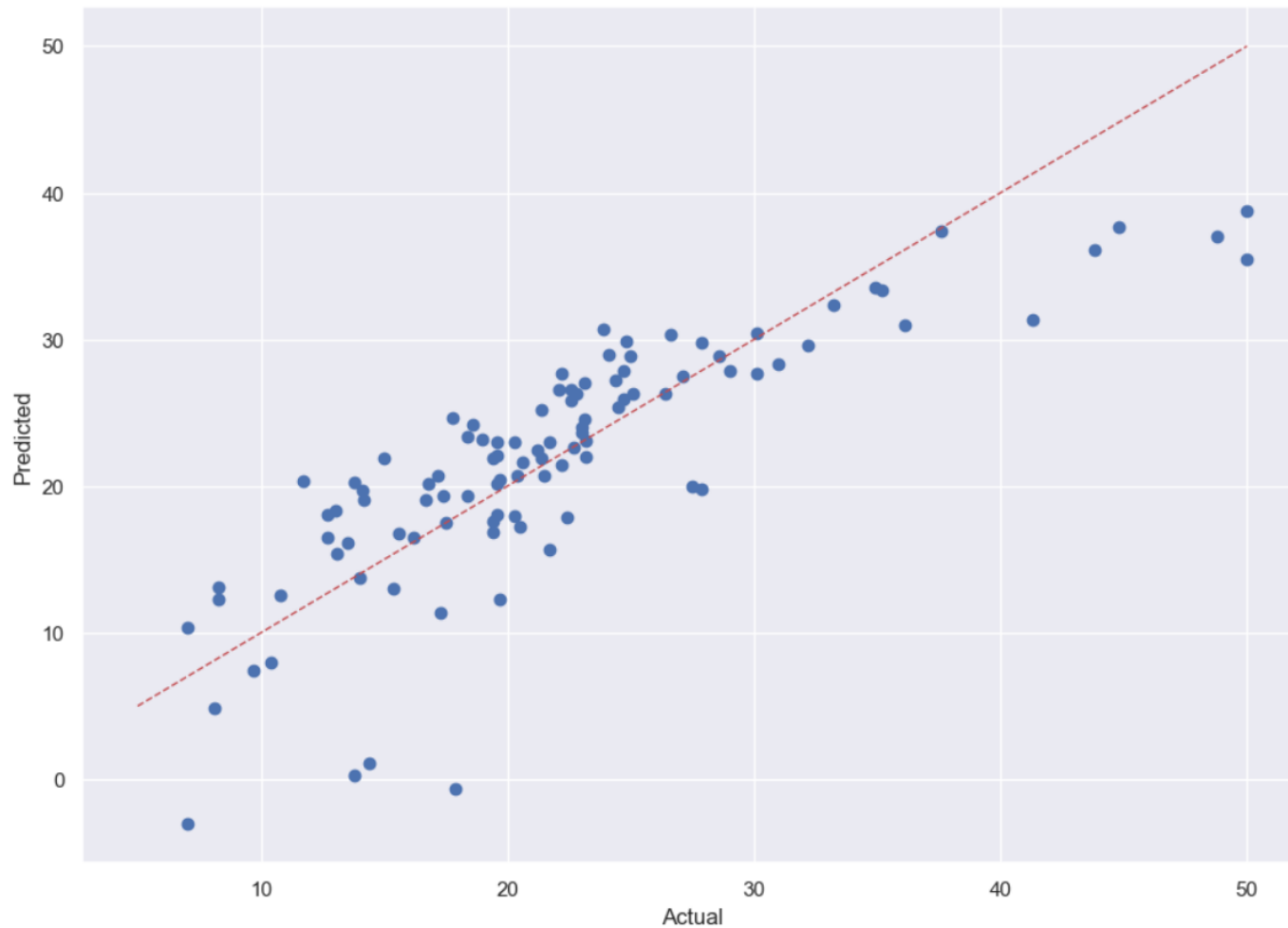
```
lin_model.intercept_
```

```
2.73624034260661
```



[실습] Boston 집값 예측

```
fig, ax = plt.subplots()
ax.scatter(Y_test, y_test_predict)
ax.plot([Y.min(), Y.max()], [Y.min(), Y.max()], 'r--', lw=1)
ax.set_xlabel('Actual')
ax.set_ylabel("Predicted")
plt.show()
```





감사합니다

Q & A

부산대학교 전기컴퓨터공학부
부산대학교 사물인터넷 연구센터장
부산대 블록체인 플랫폼 연구센터장
부산대 융합보안대학원 책임교수

김호원

howonkim@pusan.ac.kr