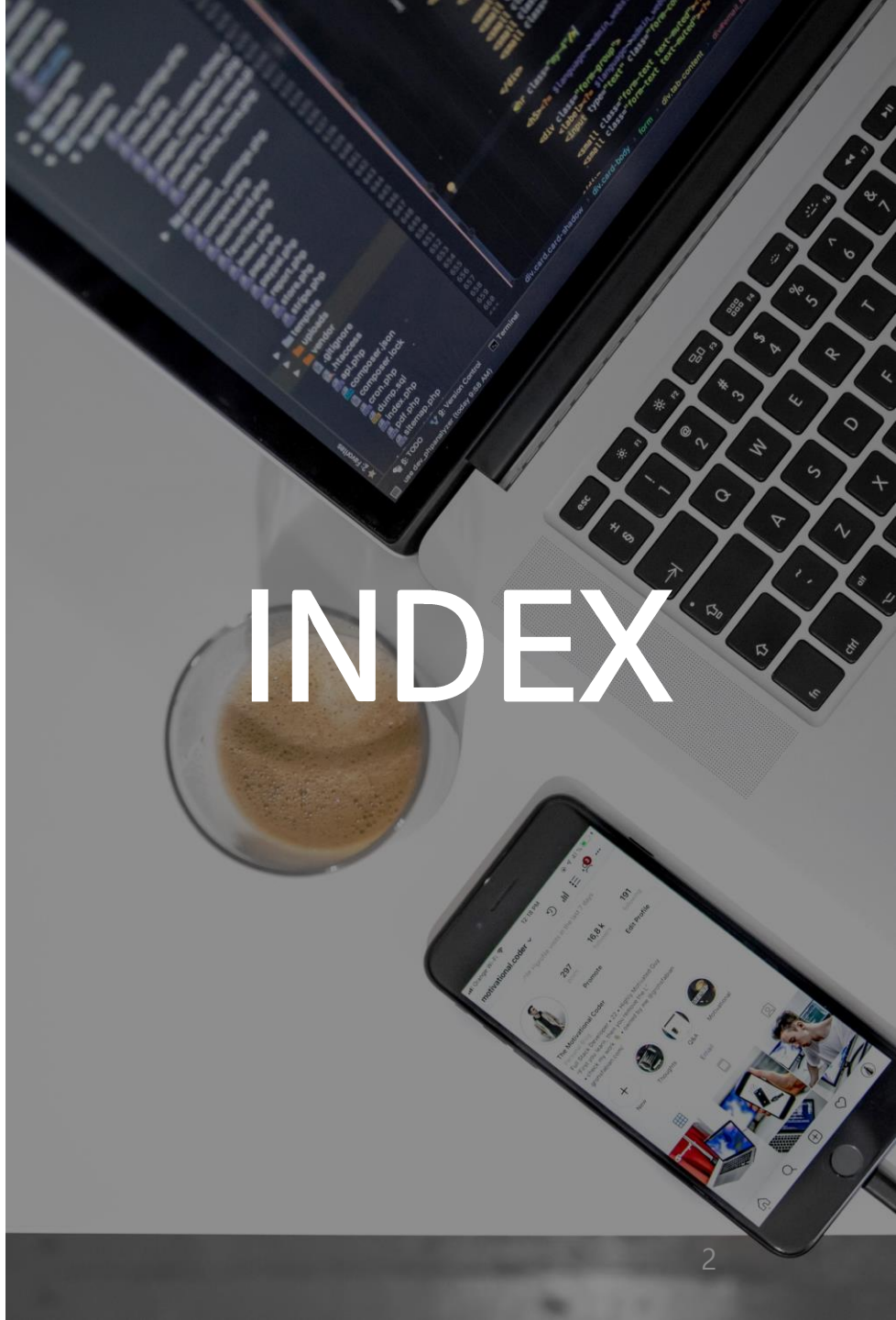
HAJ Lecture

Chapter 1. 한눈에 보는 머신러닝
Chapter 2. 머신러닝 프로젝트 처음부터 끝까지

Hands-On Machine Learning
with Scikit-Learn, Keras & TensorFlow

INDEX



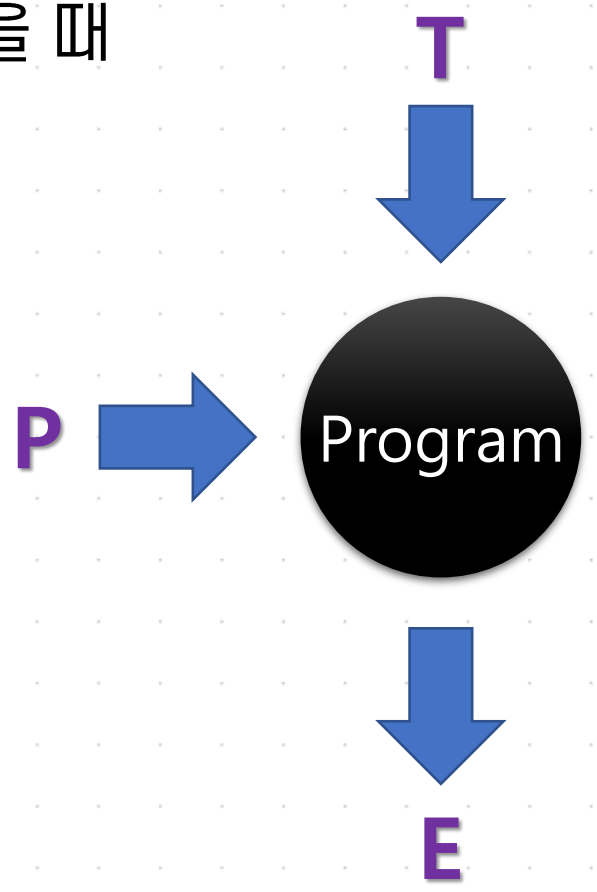
1.1 머신러닝이란?

명시적인 프로그래밍 없이 컴퓨터가 학습하는 능력을 갖추게 하는 연구 분야

어떤 작업 T에 대한 컴퓨터 프로그램의 성능을 P로 측정했을 때
경험 E로 인해 성능이 향상됐다면,

이 컴퓨터 프로그램은 작업 T와 성능 측정 P에 대해
경험 E로 학습한 것이다.

(공학적 정의)



1.1 머신러닝이란?

Training Set
시스템이 학습하는 데 사용하는 샘플

Training Instance
각각의 훈련 데이터들을 일컫는 말(또는, sample이라고도 함)

Accuracy
정확도, 성능 측정의 척도
e.g. 스팸 판정 문제에서, 정확히 분류된 메일의 비율 P

1.2 왜 머신러닝을 사용하는가?

1) 기존의 솔루션으로는 복잡하거나 알려진 정해가 없는 문제

e.g. NLP(Natural Language Processing)

2) 머신러닝을 통해 학습한 내용을 조사

데이터마이닝(data mining)

: 머신러닝 기술을 적용하여 대용량의 데이터를 분석하여 겉으로는 보이지 않던 패턴을 발견해내는 기술

1.4 머신러닝 시스템의 종류

[1] 지도 학습 vs. 비지도 학습

→ 기준: 학습하는 동안의 감독(supervision)의 형태나 정보량

지도 학습(supervised learning)

훈련 데이터에 label이라는 답을 붙이는 과정이 포함됨

(1) 분류(classification)

e.g. 스팸필터 → spam? OR Non-spam?

(2) 회귀(regression)

예측변수라 부르는 특성(feature)을 사용해 타겟(target) 수치를 예측

e.g. x : (주행거리, 연식, 브랜드, …) → y : (중고 자동차의 가격)

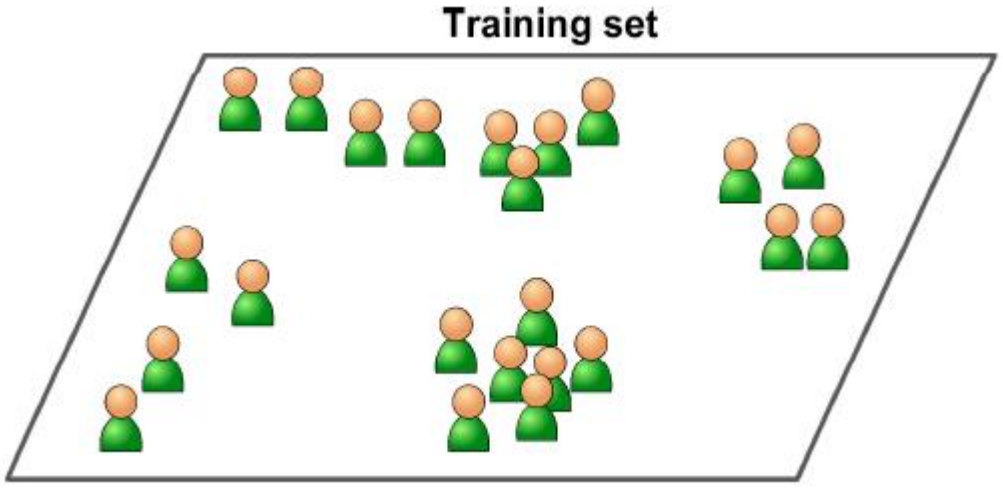
1.4 머신러닝 시스템의 종류

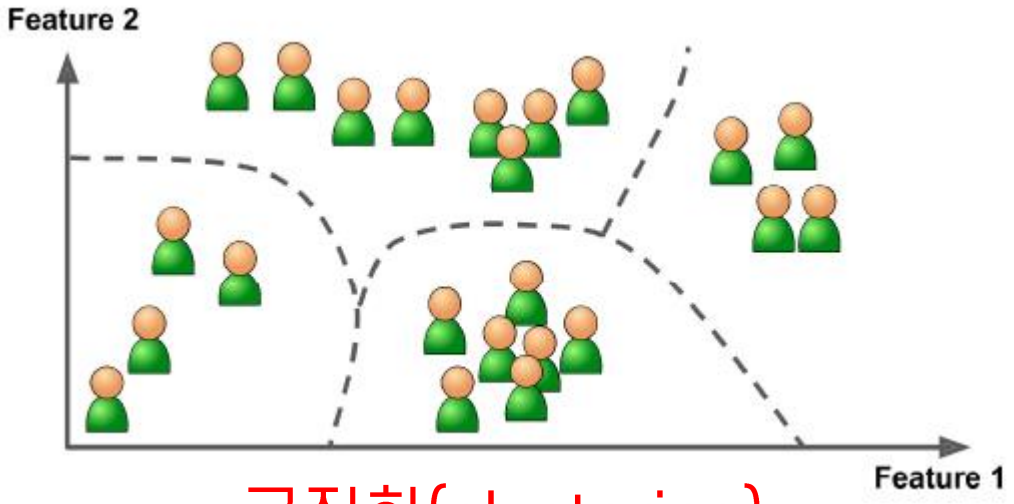
[1] 지도 학습 vs. 비지도 학습

→ 기준: 학습하는 동안의 감독(supervision)의 형태나 정보량

비지도 학습(unsupervised learning)

훈련 데이터에 label이 존재하지 않는다. → 시스템은 아무런 도움없이 학습함





군집화(clustering)

1.4 머신러닝 시스템의 종류

[1] 지도 학습 vs. 비지도 학습

→ 기준: 학습하는 동안의 감독(supervision)의 형태나 정보량

준지도 학습(semisupervised learning) → semi: “중간”

일부만 label이 있는 데이터를 다루는 것

지도학습 + 비지도학습

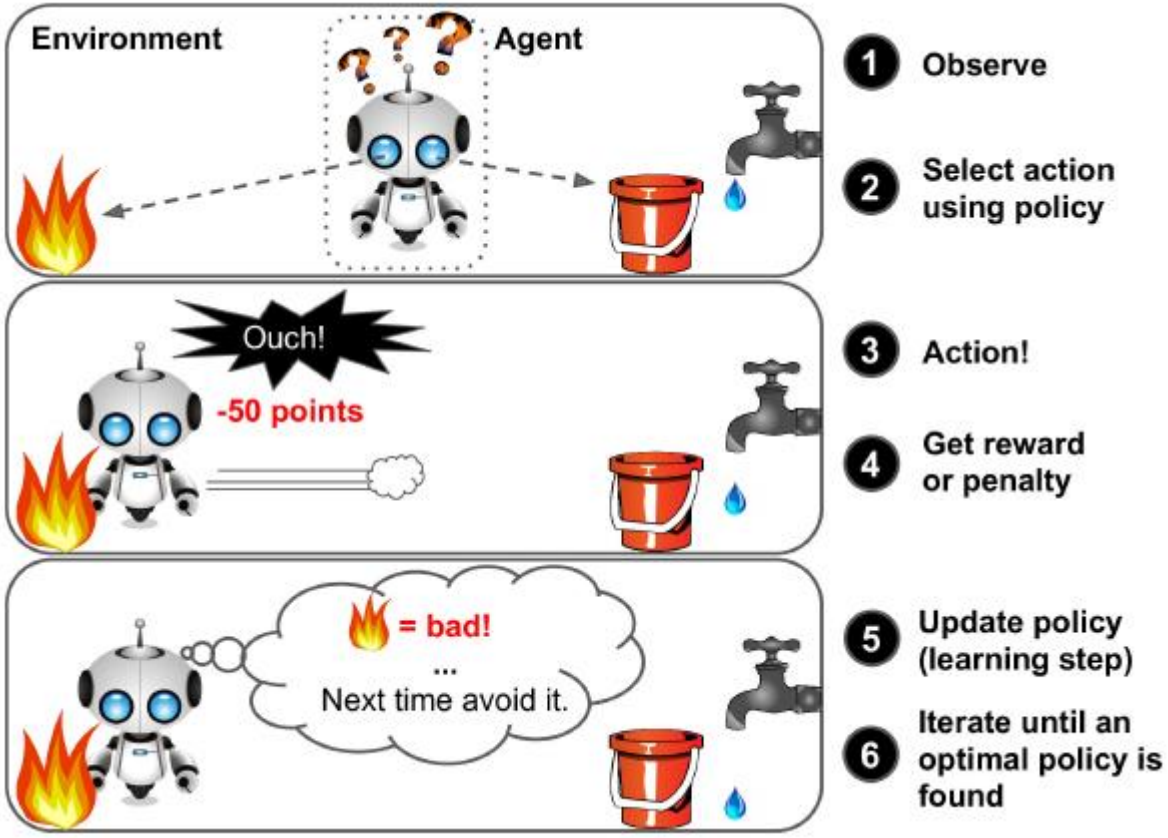
ex) 구글 포토 호스팅 서비스: 사진 속의 얼굴을 각자 다른 사람으로 구별/인식

사람A는 1,5,6번 사진에, 사람B는 1,2,3번 사진에… -> 비지도학습(군집)

사람마다 label을 붙인다. -> 지도학습(분류)

1.4 머신러닝 시스템의 종류

강화 학습(reinforcement learning)



에이전트(agent)

학습하는 시스템

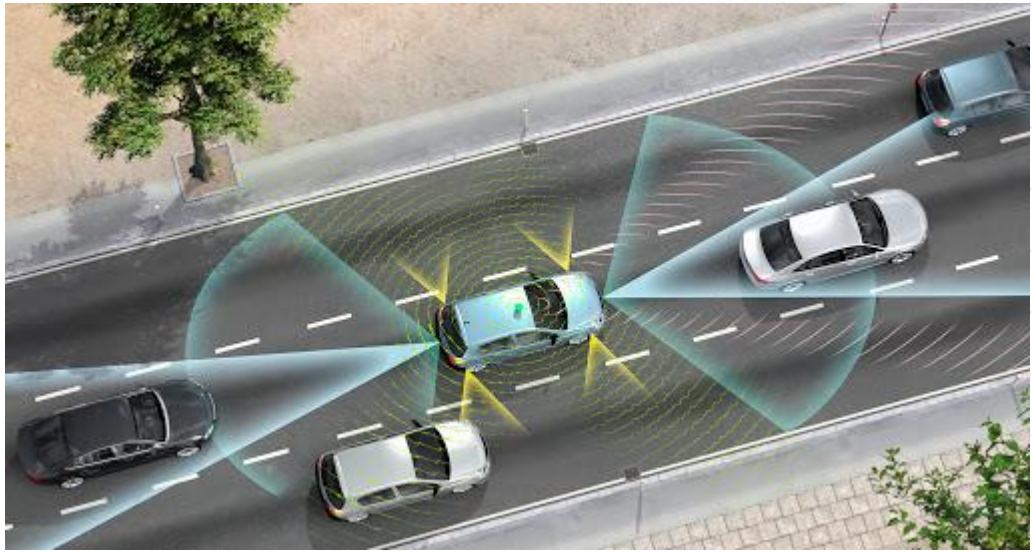
환경을 관찰하여 행동을 실행

결과: 보상(reward) - positive

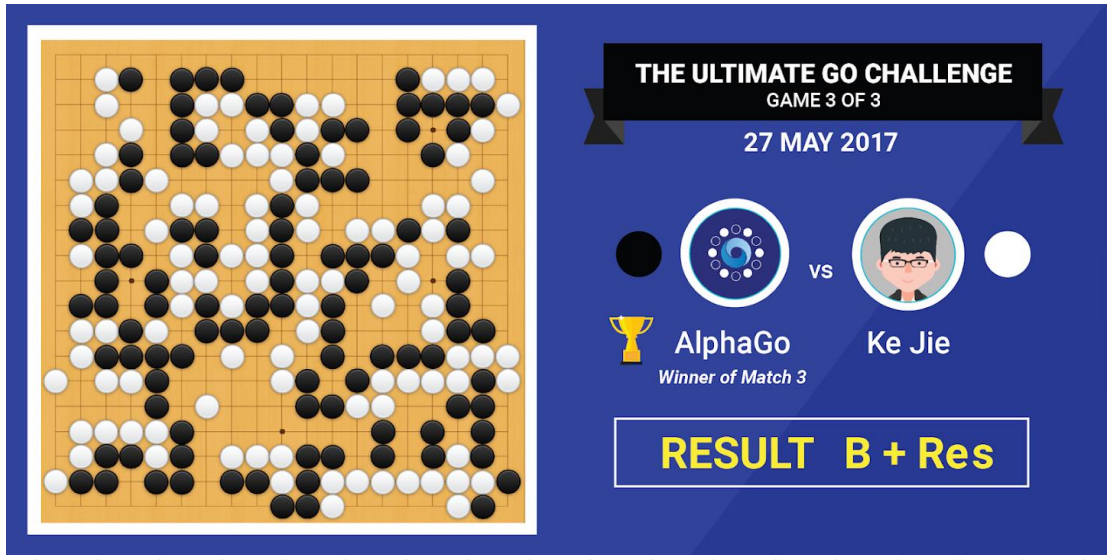
벌점(penalty) - negative

1.4 머신러닝 시스템의 종류

강화 학습(reinforcement learning)



자율 주행 기술



알파고(AlphaGo) from DeepMind

1.4 머신러닝 시스템의 종류

[2] 배치 학습 vs. 온라인 학습

→ 기준: 입력 데이터의 스트림으로부터 점진적으로 학습할 수 있는가?

배치 학습(batch learning)

시스템이 점진적으로 학습할 수 없는 경우 → 모든 데이터를 사용하여야 함

- 새로운 데이터를 학습하려면 전체 데이터를 사용하여
  처음부터 다시 훈련을 수행해야 한다.
- 시스템이 빠르게 변화하는 특성을 가져야하는 경우에는 부적절함

* batch: 모델을 한번 update시킬 때 사용하는 샘플들의 묶음

1.4 머신러닝 시스템의 종류

[2] 배치 학습 vs. 온라인 학습

→ 기준: 입력 데이터의 스트림으로부터 점진적으로 학습할 수 있는가?

온라인 학습(online learning)

시스템이 점진적으로 학습할 수 있는 경우

- 매 학습단계가 빠르고 비용이 적게들어 데이터가 도착하는대로 학습함
- 시스템이 빠르게 변화하는 특성을 가져야하는 경우에 적절한 방법
- 학습률(learning rate): 변화하는 데이터에 얼마나 빠르게 적응하는지에 대한 척도

  i) 학습률이 높다면? 새 데이터에 빠르게 적응, but 예전 데이터를 금방 잊어버림

  ii) 학습률이 낮다면? 시스템의 학습이 더디게 일어남

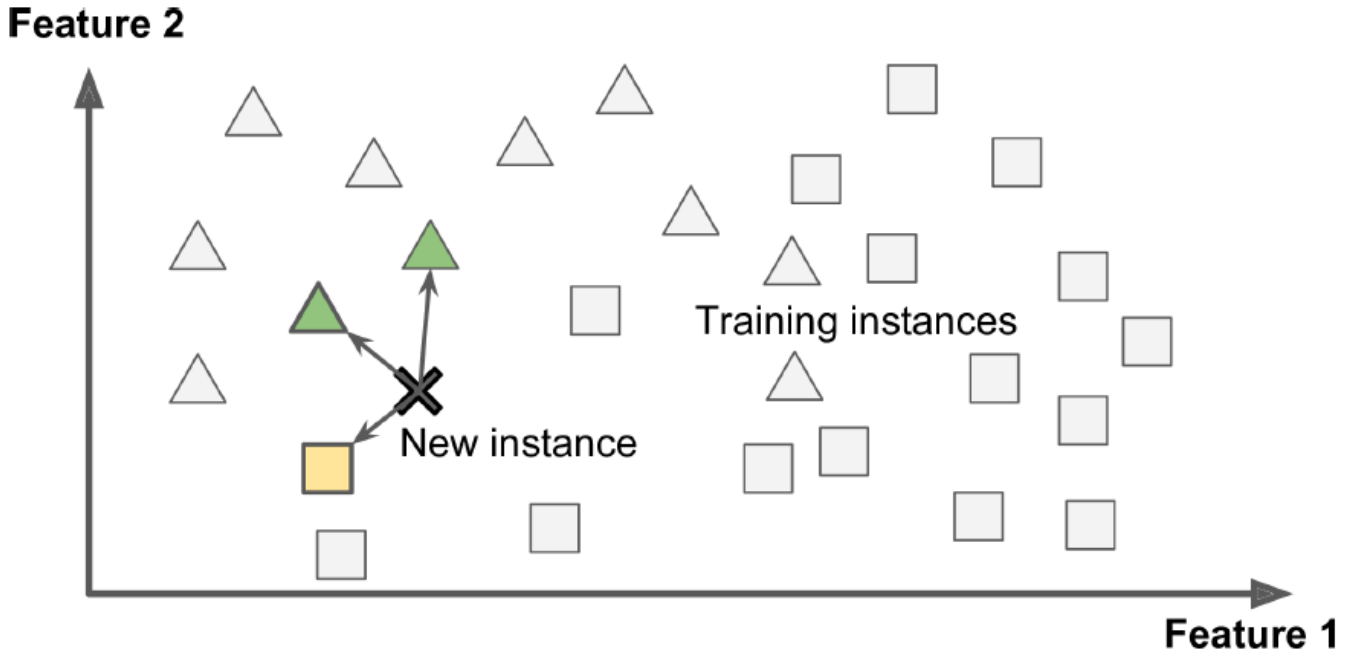
1.4 머신러닝 시스템의 종류

[3] 사례 기반 학습 vs. 모델 기반 학습

→ 기준: 머신러닝 시스템이 어떻게 일반화(generalize)되는가

사례 기반 학습(instance-based learning)

시스템이 훈련 샘플을 기억함으로써 학습하는 것



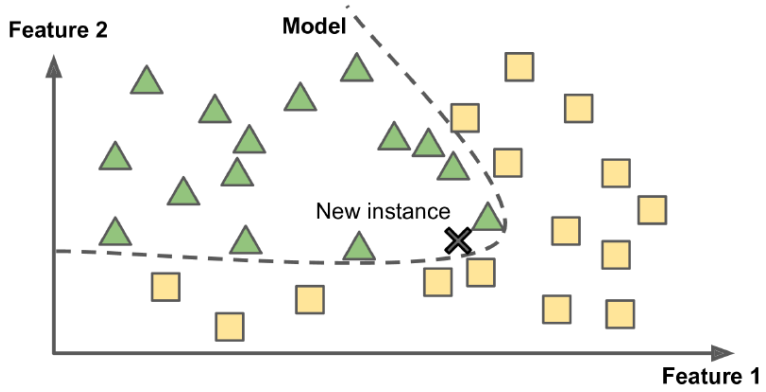
1.4 머신러닝 시스템의 종류

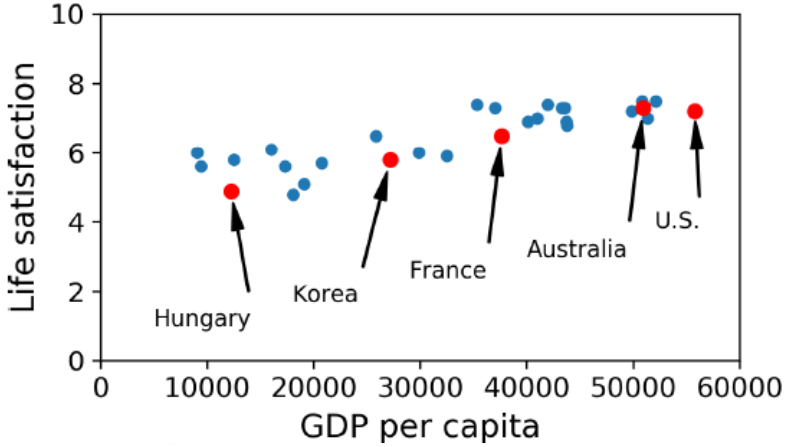
[3] 사례 기반 학습 vs. 모델 기반 학습

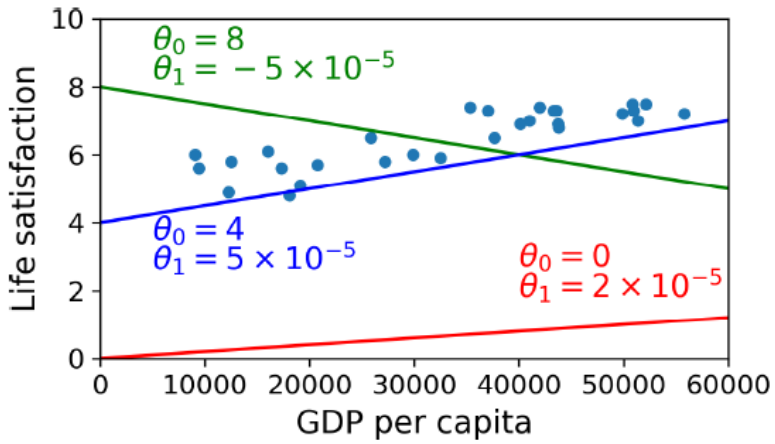
→ 기준: 머신러닝 시스템이 어떻게 일반화(generalize)되는가

모델 기반 학습(model-based learning)

훈련 샘플들의 모델을 만들어 예측(prediction)에 사용하는 것







선형 모델(linear model)

모델 파라미터(model parameter): θ_0, θ_1

1.4 머신러닝 시스템의 종류

모델 기반 학습(model-based learning)

훈련 샘플들의 모델을 만들어 예측(prediction)에 사용하는 것

- 성능 측정의 지표

  i) 얼마나 모델이 좋은가? → 효용 함수(utility function)

  ii) 얼마나 모델이 나쁜가? → 비용 함수(cost function)

  in 선형모델, “점과 직선 사이의 거리 최소화“ → cost function

- 훈련

  알고리즘에 훈련 데이터를 공급하여 데이터에 가장 잘 맞는

  모델의 파라미터를 찾아가는 과정

1.4 머신러닝 시스템의 종류

모델 기반 학습(model-based learning)

데이터를 분석한다.

모델을 선택한다.

훈련 데이터로 모델을 훈련시킨다.(=비용함수의 최소화되는 모델 파라미터 찾기)

새로운 데이터에 모델을 적용하여 예측을 수행

1.4 머신러닝 시스템의 종류

모델 기반 학습(model-based learning)

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn.linear_model

# Load the data
oecd_bli = pd.read_csv("oecd_bli_2015.csv", thousands=',')
gdp_per_capita = pd.read_csv("gdp_per_capita.csv",thousands=',',delimiter='\t',
                             encoding='latin1', na_values="n/a")

# Prepare the data
country_stats = prepare_country_stats(oecd_bli, gdp_per_capita)
X = np.c_[country_stats["GDP per capita"]]
y = np.c_[country_stats["Life satisfaction"]]

# Visualize the data
country_stats.plot(kind='scatter', x="GDP per capita", y='Life satisfaction')
plt.show()

# Select a linear model
model = sklearn.linear_model.LinearRegression()

# Train the model
model.fit(X, y)

# Make a prediction for Cyprus
X_new = [[22587]]  # Cyprus' GDP per capita
print(model.predict(X_new)) # outputs [[ 5.96242338]]
```

데이터를 분석한다.

모델을 선택한다.

훈련 데이터로 모델을 훈련시킨다.

새로운 데이터에 모델을 적용하여 예측을 수행

1.6 테스트와 검증

학습시킨 모델이 새로운 데이터가 들어왔을 때 얼마나 잘 적용할 수 있을까?

- 훈련 데이터를 훈련 세트(trainset)/데이터 세트(testset)로 나누는 것

  i) 훈련 세트(trainset) : 이를 이용하여 모델을 훈련

  ii) 테스트 세트(testset): 이를 이용하여 모델을 테스트

- 일반화 오차(generalized error)

  새로운 샘플에 대한 오류 비율

  → 이전에 학습한 적이 없는 새로운 데이터에 얼마나 잘 작동하는지의 척도

2.1 실제 데이터로 작업하기

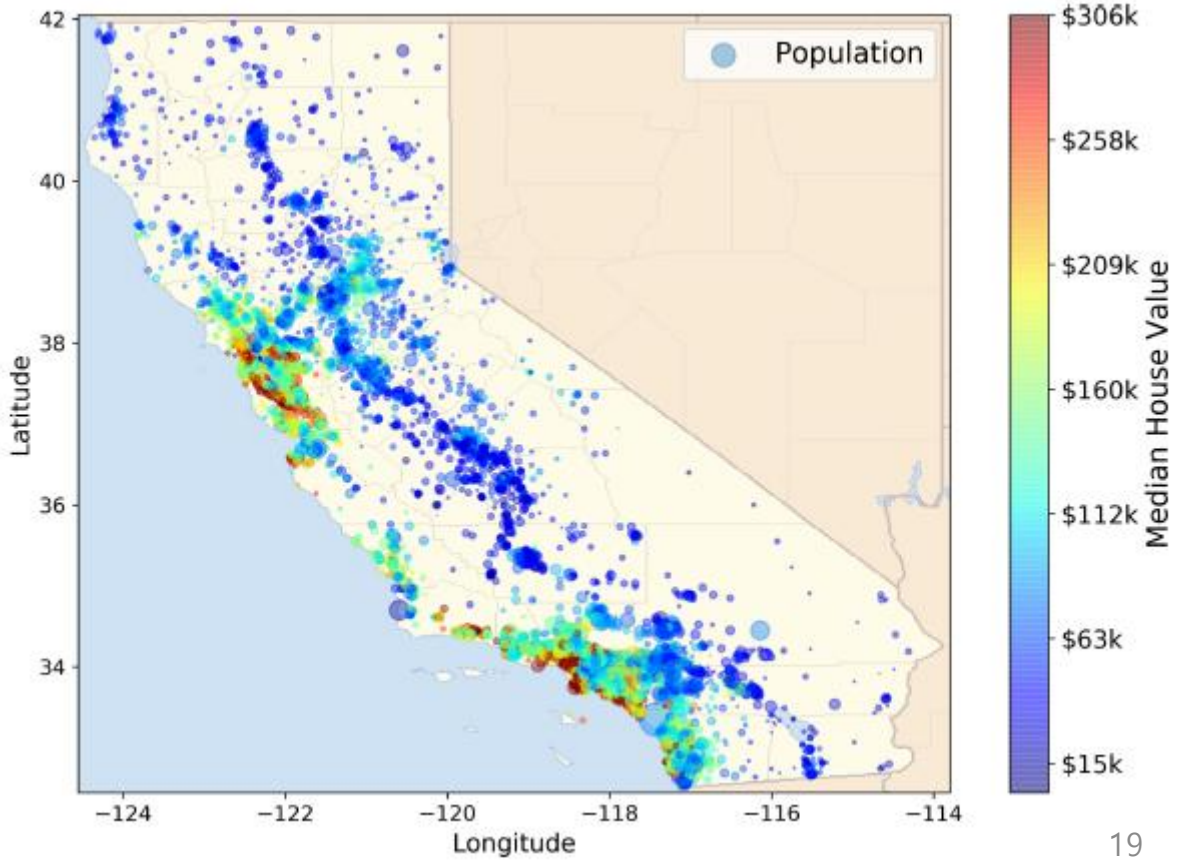
이번 수업에서는...

인공적으로 만들어진 데이터셋보다는, 실제 데이터로 실험해볼 것입니다.

→ 캘리포니아 주택 가격

(California Housing Prices)

데이터셋 *in StatLib*



2.2 큰 그림 보기

이번 프로젝트의 목적은?

캘리포니아 인구조사 데이터를 사용하여

캘리포니아의 주택 가격 모델을 만들어보자.

- 중간주택가격을 포함하는 각각의 샘플 = Label이 있는 훈련 샘플
  → 이는 지도학습에 해당한다.
- 주택의 가격을 예측해야 한다. → 특정 값을 예측 → 회귀(regression)
- 예측에 사용하는 특성이 여러 개(x값 여러 개) → 다중회귀(multiple regression)
- 주택가격 1개만 예측(y값 1개) → 단변량 회귀(univariate regression)

2.2 큰 그림 보기

성능 측정의 지표

1) 평균 제곱근 오차(RMSE; root mean square error)

$$RMSE(X,h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(x^i) - y^i)^2}$$

2) 평균 절대 오차(MAE; mean absolute error)

$$MAE(X,h) = \frac{1}{m}\sum_{i=1}^{M}|h(x^i) - y^i|$$

2.3 데이터 가져오기

CSV 파일 (comma-separated value)

1) pandas로 데이터 불러오기

```
import pandas as pd

def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
```

2) 데이터프레임에 head( ) 메서드 이용

```
In [5]: housing = load_housing_data()
        housing.head()
```

3) 숫자형 특성을 알아보기 위한 describe( ) 메서드 이용

```
In [8]: housing.describe()
```

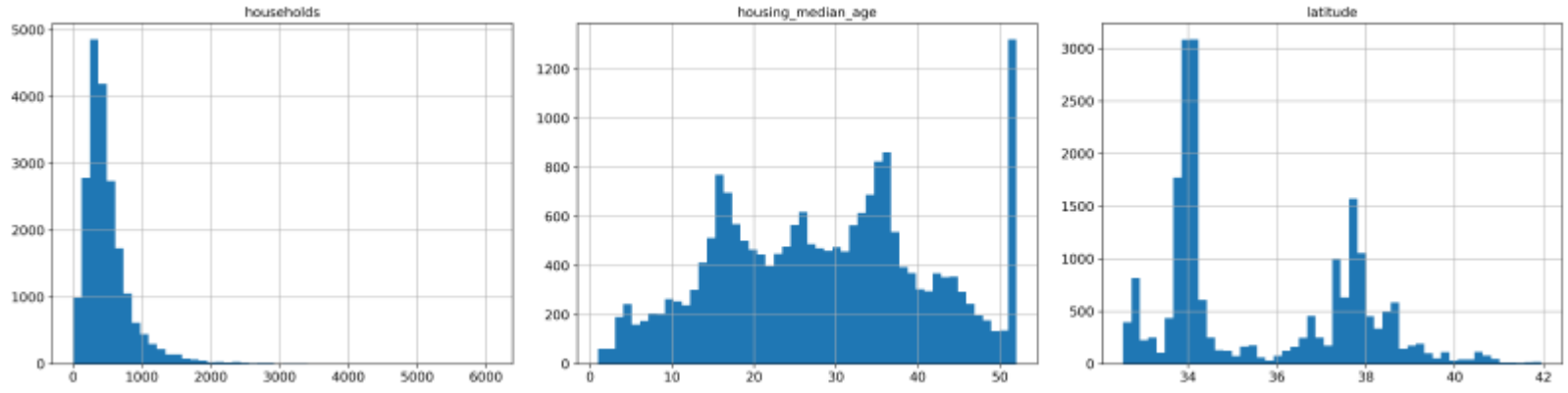
2.3 데이터 가져오기

CSV 파일 (comma-separated value)

4) 히스토그램(histogram)

```
%matplotlib inline    # 주피터 노트북 환경에서 입력
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))
plt.show()
```

- 데이터의 형태/분포를 빠르게 검토할 수 있다.
- 주어진 값의 범위(수평축) + 그 범위에 속한 샘플 수(수직축)



2.3 데이터 가져오기

테스트 세트 만들기

무작위로 어떤 샘플을 선택해서 전체 데이터셋의 20% 정도를 떼어놓는 것을 해보겠습니다.

```
import numpy as np

def split_train_test(data, test_ratio):
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(len(data) * test_ratio)
    test_indices = shuffled_indices[:test_set_size]
    train_indices = shuffled_indices[test_set_size:]
    return data.iloc[train_indices], data.iloc[test_indices]

>>> train_set, test_set = split_train_test(housing, 0.2)
>>> len(train_set)
16512
>>> len(test_set)
4128
```

2.3 데이터 가져오기

테스트 세트 만들기

train_test_split

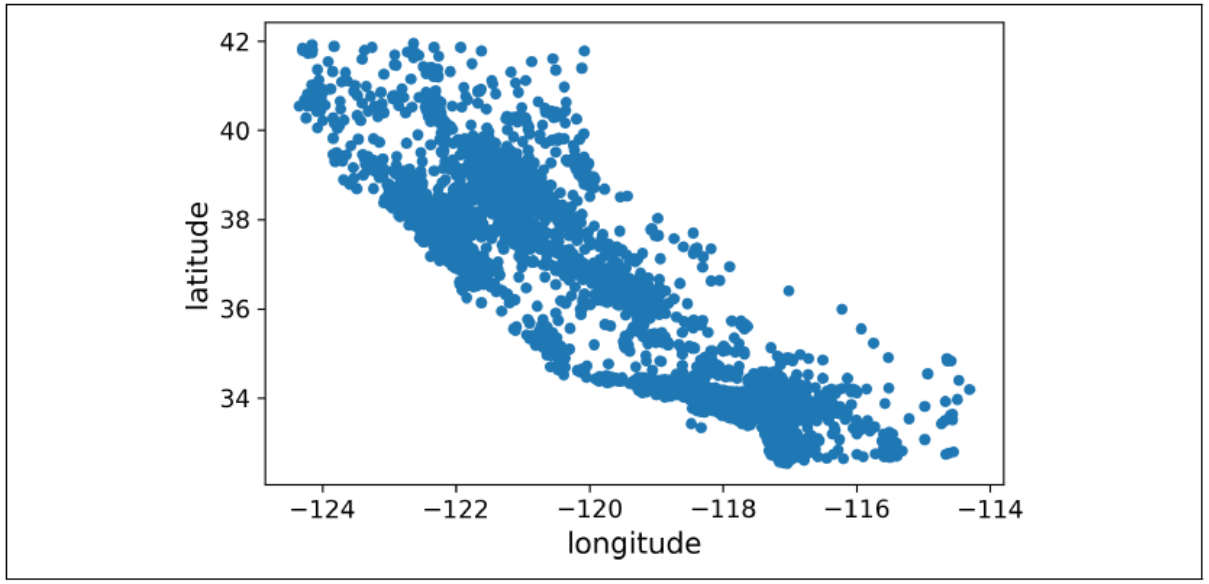
- 난수 초기값을 설정할 수 있다. --> random_state
- 행의 개수가 같은 여러 개의 데이터셋을 넘겨서 같은 인덱스를 기반으로 나눌 수 있다.
  (예를 들어 데이터프레임이 label을 따라 여러 개로 나뉘어 있을 때 유용하다.)

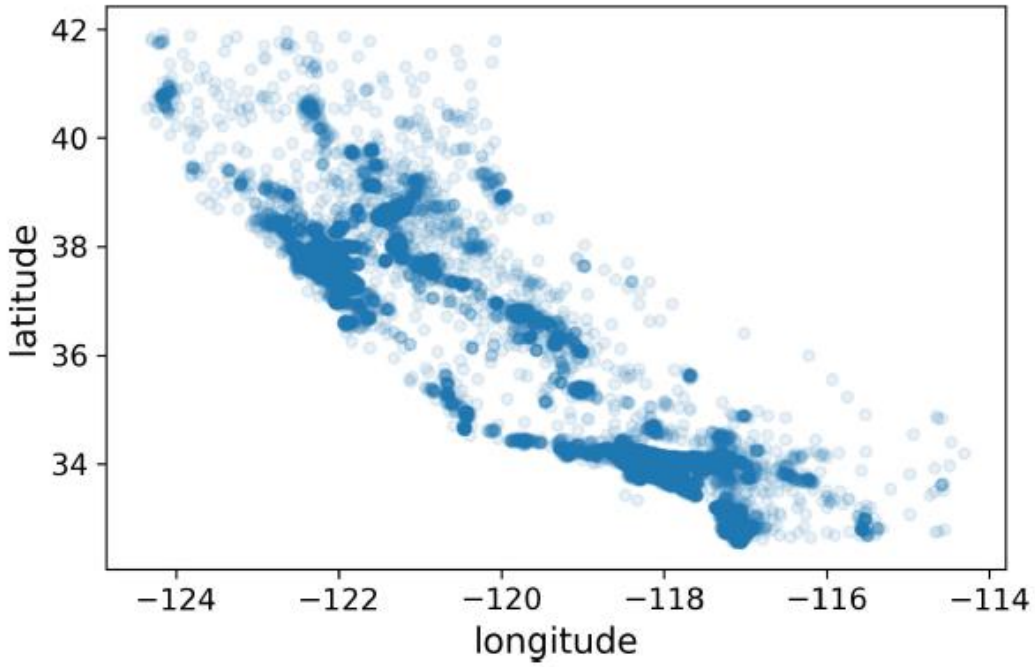
```
from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
```

2.4 데이터 이해를 위한 탐색과 시각화

산점도 데이터 시각화

```
housing.plot(kind="scatter", x="longitude", y="latitude")

housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.1)
```





2.4 데이터 이해를 위한 탐색과 시각화

표준 상관계수(standard correlation coefficient)

두 변수 사이의 통계적 관계를 표현하기 위해 특정한 상관 관계의 정도를 수치적으로 나타낸 계수

* 상관계수의 범위: [-1, 1]

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

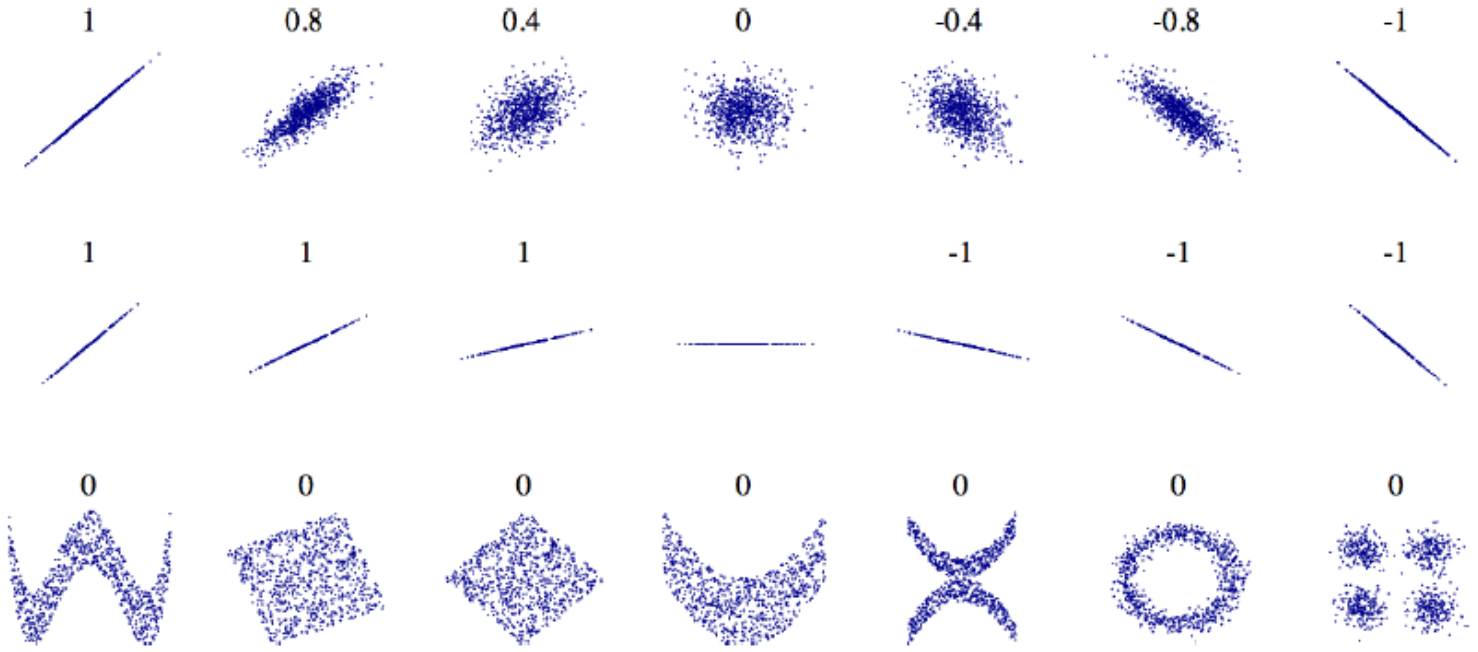
```
corr_matrix = housing.corr()
```

```
>>> corr_matrix["median_house_value"].sort_values(ascending=False)
median_house_value    1.000000
median_income         0.687170
total_rooms           0.135231
housing_median_age    0.114220
households            0.064702
total_bedrooms        0.047865
population           -0.026699
longitude            -0.047279
latitude             -0.142826
Name: median_house_value, dtype: float64
```

2.4 데이터 이해를 위한 탐색과 시각화

표준 상관계수(standard correlation coefficient)

- 계수가 -1에 가깝다 → 강한 음의 상관관계
- 계수가 0에 가깝다 → 선형적인 상관관계가 없다
- 계수가 +1에 가깝다 → 강한 양의 상관관계



2.5 머신러닝 알고리즘을 위한 데이터 준비

데이터 정제

대부분의 머신러닝 알고리즘은 누락된 특성을 다루지 못한다.

→ 이를 처리할 함수가 필요하고 크게 3가지의 방법이 존재한다.

- 해당 구역을 제거한다.
- 전체 특성을 삭제한다.
- 어떤 값으로 채운다.(0, 평균, 중간값 …)

```
housing.dropna(subset=["total_bedrooms"])    # option 1
housing.drop("total_bedrooms", axis=1)       # option 2
median = housing["total_bedrooms"].median()  # option 3
housing["total_bedrooms"].fillna(median, inplace=True)
```

2.5 머신러닝 알고리즘을 위한 데이터 준비

데이터 정제

Scikit-learn의 SimpleImputer 객체

1) SimpleImputer 객체를 생성(누락된 값을 특성의 중간값을 대체한다고 가정)

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy="median")
```

2) 텍스트 특성은 제외한 복사본을 생성한다.(수치형 특성에서만 중간값이 계산되어야 한다.)

```
housing_num = housing.drop("ocean_proximity", axis=1)
```

3) imputer 객체의 fit( ) 메서드를 이용하여 훈련 데이터에 적용한다.

```
imputer.fit(housing_num)
```

4) 학습된 imputer 객체를 이용하여 훈련 세트에서 누락된 값을 학습한 중간값으로 바꾼다.

```
X = imputer.transform(housing_num)
```

2.5 머신러닝 알고리즘을 위한 데이터 준비

텍스트와 범주형 특성 다루기

```
>>> housing_cat = housing[["ocean_proximity"]]
>>> housing_cat.head(10)
       ocean_proximity
17606        <1H OCEAN
18632        <1H OCEAN
14650       NEAR OCEAN
3230            INLAND
3555         <1H OCEAN
19480           INLAND
8879         <1H OCEAN
13685           INLAND
4937         <1H OCEAN
4861         <1H OCEAN
```

- 이전의 예시에서 텍스트 형식의 데이터는 ocean_proximity 1개였다.
- 데이터를 보니, 몇 종류의 값 중 하나를 갖는다.
- Idea: “이를 수치형 데이터로 변환할 수 있지 않을까?”

```
>>> from sklearn.preprocessing import OrdinalEncoder
>>> ordinal_encoder = OrdinalEncoder()
>>> housing_cat_encoded = ordinal_encoder.fit_transform(housing_cat)
>>> housing_cat_encoded[:10]
```

```
array([[0.],
       [0.],
       [4.],
       [1.],
       [0.],
       [1.],
       [0.],
       [1.],
       [0.],
       [0.]])
```

2.5 머신러닝 알고리즘을 위한 데이터 준비

텍스트와 범주형 특성 다루기

categories_ 인스턴스 변수를 사용해 카테고리 목록을 얻을 수 있다.

```
>>> ordinal_encoder.categories_
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
       dtype=object)]
```

원-핫 인코딩(one-hot encoding)

한 특성만 1(hot)이고, 나머지는 모두 0으로 인코딩하는 것

e.g. “INLAND”만 1, 나머지는 모두 0

```
>>> from sklearn.preprocessing import OneHotEncoder
>>> cat_encoder = OneHotEncoder()
>>> housing_cat_1hot = cat_encoder.fit_transform(housing_cat)
>>> housing_cat_1hot
<16512x5 sparse matrix of type '<class 'numpy.float64'>'
  with 16512 stored elements in Compressed Sparse Row format>
```

2.5 머신러닝 알고리즘을 위한 데이터 준비

직접 변환기 만들기

특별한 정제 작업이나 어떤 특성들을 조합하는
등의 작업을 위해
직접 변환기를 설계해야 하는 경우가 존재한다.

→ 사이킷런은 duck typing을 지원한다.

→ fit( ), transform( ), fit_transform( ) 메서드를 구현한 객체를 생성

```
from sklearn.base import BaseEstimator, TransformerMixin

rooms_ix, bedrooms_ix, population_ix, households_ix = 3, 4, 5, 6

class CombinedAttributesAdder(BaseEstimator, TransformerMixin):
    def __init__(self, add_bedrooms_per_room = True): # no *args or **kargs
        self.add_bedrooms_per_room = add_bedrooms_per_room
    def fit(self, X, y=None):
        return self  # nothing else to do
    def transform(self, X, y=None):
        rooms_per_household = X[:, rooms_ix] / X[:, households_ix]
        population_per_household = X[:, population_ix] / X[:, households_ix]
        if self.add_bedrooms_per_room:
            bedrooms_per_room = X[:, bedrooms_ix] / X[:, rooms_ix]
            return np.c_[X, rooms_per_household, population_per_household,
                         bedrooms_per_room]
        else:
            return np.c_[X, rooms_per_household, population_per_household]

attr_adder = CombinedAttributesAdder(add_bedrooms_per_room=False)
housing_extra_attribs = attr_adder.transform(housing.values)
```

2.6 모델 선택과 훈련

2.6.1 훈련 세트에서 훈련하고 평가하기

선형 회귀 모델을 훈련하는 코드이다. LinearRegression 객체를 이용하여 만들 수 있다.

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(housing_prepared, housing_labels)
```

몇 개의 샘플에 대해 적용한 결과는 아래와 같다.

```
>>> some_data = housing.iloc[:5]
>>> some_labels = housing_labels.iloc[:5]
>>> some_data_prepared = full_pipeline.transform(some_data)
>>> print("Predictions:", lin_reg.predict(some_data_prepared))
Predictions: [ 210644.6045  317768.8069  210956.4333  59218.9888  189747.5584]
>>> print("Labels:", list(some_labels))
Labels: [286600.0, 340600.0, 196900.0, 46300.0, 254500.0]
```

2.6 모델 선택과 훈련

2.6.1 훈련 세트에서 훈련하고 평가하기

mean_square_error 함수를 이용하여 만든 모델의 RMSE를 측정해보자.

```
>>> from sklearn.metrics import mean_squared_error
>>> housing_predictions = lin_reg.predict(housing_prepared)
>>> lin_mse = mean_squared_error(housing_labels, housing_predictions)
>>> lin_rmse = np.sqrt(lin_mse)
>>> lin_rmse
68628.19819848922
```

→ 과소적합(under-fitting)의 사례

→ 더 강력을 모델을 선택 or 알고리즘에 더 좋은 특성을 주입

2.6 모델 선택과 훈련

2.6.1 훈련 세트에서 훈련하고 평가하기

DecisionTreeRegressor 모델을 이용해보자.

이는 데이터에서 복잡한 비선형 관계를 찾는 것이 가능하다.

```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor()
tree_reg.fit(housing_prepared, housing_labels)
```

```
>>> housing_predictions = tree_reg.predict(housing_prepared)
>>> tree_mse = mean_squared_error(housing_labels, housing_predictions)
>>> tree_rmse = np.sqrt(tree_mse)
>>> tree_rmse
0.0
```

→ 과대적합(over-fitting)의 사례

→ 훈련 세트의 일부분으로 훈련을 하고, 다른 일부분은 모델 검증에 사용해야 한다.

2.6 모델 선택과 훈련

2.6.2 교차 검증을 사용한 평가

결정 트리 모델을 평가하는 방법에 대해 생각해보자.

k-겹 교차 검증(k-fold cross-validation)

훈련 세트를 fold라 불리는 k개의 서브넷으로 무작위 분할

→ 결정 트리 모델을 k번 훈련하고 평가

→ 매번 다른 fold를 선택하여 평가에 사용하고, 나머지 k-1개의 fold는 훈련에 사용

→ k개의 평가 점수가 담긴 배열이 결과로 도출된다.

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(tree_reg, housing_prepared, housing_labels,
                         scoring="neg_mean_squared_error", cv=10)
tree_rmse_scores = np.sqrt(-scores)
```

```
Scores: [70194.33680785 66855.16363941 72432.58244769 70758.73896782
 71115.88230639 75585.14172901 70262.86139133 70273.6325285
 75366.87952553 71231.65726027]
Mean: 71407.68766037929
Standard deviation: 2439.4345041191004
```


Thank You