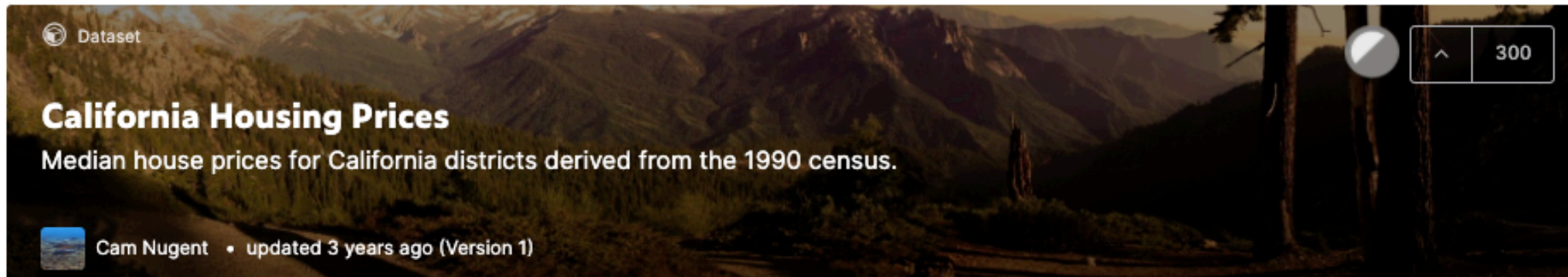


BIG DATA ANALYTICS

WEEK-14 | Application-1 Regression

**Yonsei University
Jungwon Seo**


캘리포니아 주택 가격 예측하기





California Housing Prices
Median house prices for California districts derived from the 1990 census.

Cam Nugent • updated 3 years ago (Version 1)

[Data](#) [Tasks \(1\)](#) [Notebooks \(163\)](#) [Discussion \(4\)](#) [Activity](#) [Metadata](#) [Download \(1 MB\)](#) [New Notebook](#)

 **Usability** 8.5

 **License** CC0: Public Domain

 **Tags** computer science, software, programming, social science, social issues and advocacy and 3 more

Description

Context

This is the dataset used in the second chapter of Aurélien Géron's recent book 'Hands-On Machine learning with Scikit-Learn and TensorFlow'. It serves as an excellent introduction to implementing machine learning algorithms because it requires rudimentary data cleaning, has an easily understandable list of variables and sits at an optimal size between being too toyish and too cumbersome.

The data contains information from the 1990 California census. So although it may not help you with predicting current housing prices like the Zillow Zestimate dataset, it does provide an accessible introductory dataset for teaching people about the basics of machine learning.

1. 데이터 구조 훑어 보기

- (주의) 각 행(row)은 특정 집이 아닌 지역

- Feature 확인

- longitude/latitude: 경도/위도
- housing_median_age: 주택 연식 중위값
- total_rooms: 지역의 전체 방의 수
- total_bedrooms: 지역의 전체 침실의 개수
- population: 지역의 인구
- households: 가구 수
- median_income: 중위소득
- median_house_value: 주택가격의 중위값
- ocean_proximity: 해안가와의 가까움 정도

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

1. 데이터 구조 훑어 보기

데이터 수

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude           20640 non-null float64
latitude            20640 non-null float64
housing_median_age  20640 non-null float64
total_rooms         20640 non-null float64
total_bedrooms      20433 non-null float64
population          20640 non-null float64
households          20640 non-null float64
median_income       20640 non-null float64
median_house_value  20640 non-null float64
ocean_proximity     20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

범주형 데이터에 대한 요약통계

```
housing["ocean_proximity"].value_counts()
```

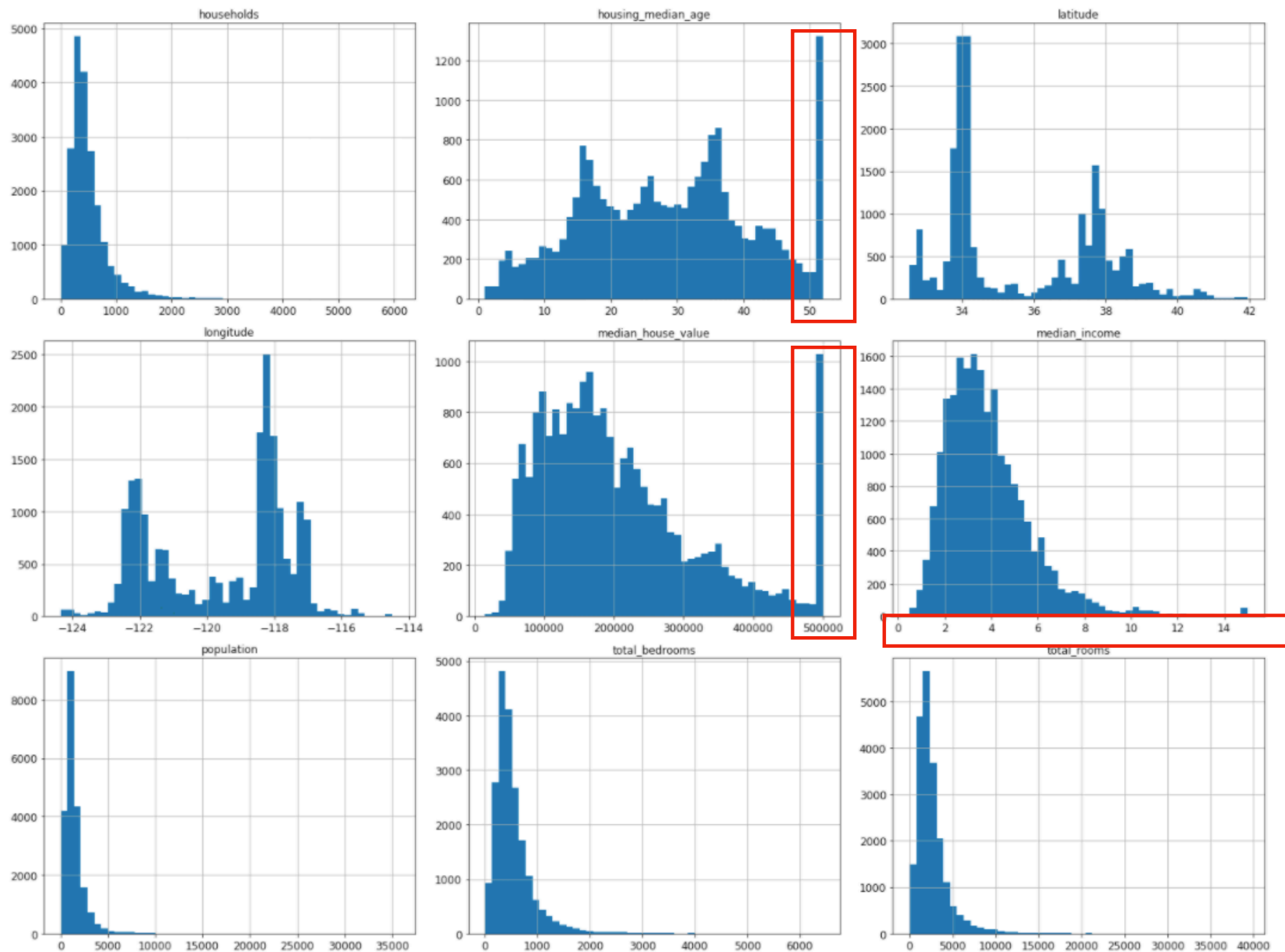
```
<1H OCEAN      9136
INLAND         6551
NEAR OCEAN     2658
NEAR BAY       2290
ISLAND          5
Name: ocean_proximity, dtype: int64
```

수치형 데이터에 대한 요약통계

```
housing.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

1. 데이터 구조 훑어 보기



각 속성별 히스토그램

1. 데이터 구조 훑어 보기

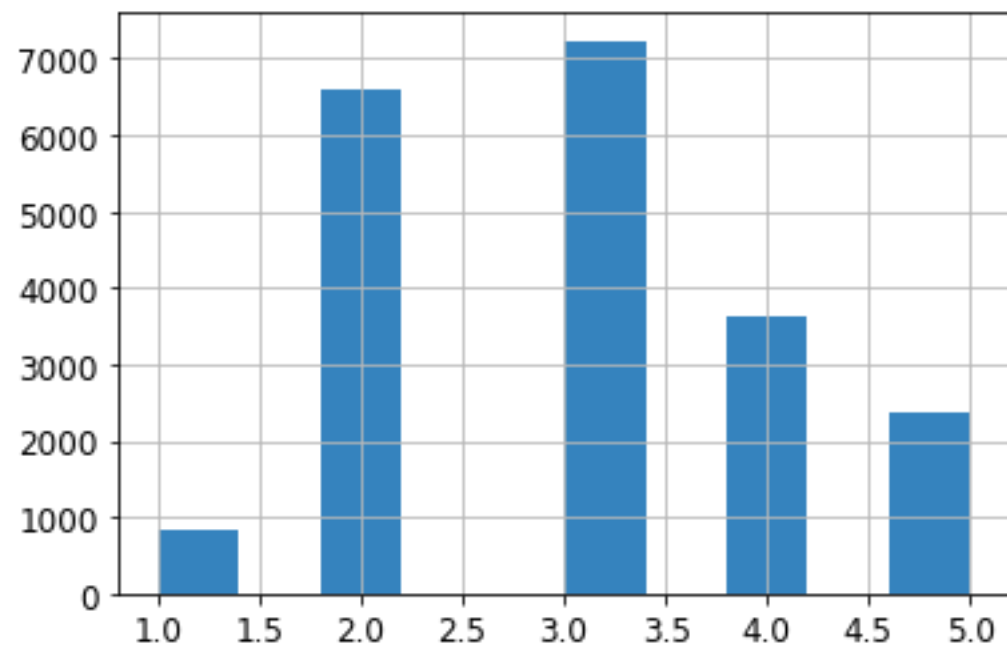
- (주의) 데이터를 더 깊게 보기전에 미리 테스트 데이터셋을 분리해야함
- 중위소득의 단위 확인
- 침실 수의 결측값 인지
- 주택 연식과 가격의 최대값이 한정되어 있는데, 이 최대값을 넘어가는 예측이 필요한지 확인!
 - 데이터셋의 주택가격의 최대값이 \$500,000인데 실제 주택 가격이 몇 백만 달러일수도 있음
 - 만약 필요하다면, 실제 값을 확보
 - 확보 할 수 없다면 이 구간을 제거

2. 테스트 세트 만들기

- 왜 테스트 세트를 만들어야 하나?
 - 일반화된 모델을 만들기 위해서는 최종 모델을 검증하기 전까지 사람도 모델도 테스트 세트에 대한 영향을 받으면 안됨
 - 자칫 잘못하면 테스트 세트에 최적화된 모델을 개발 할 수가 있음
- 전통 적인 머신러닝에서는 일반적으로 전체 데이터 셋의 20%를 테스트 세트로 가져감
 - 물론 데이터가 1억개 이렇게 있다면, 2천만개를 굳이 테스트 세트로 가져가지 않아도 충분함
- 어떻게 20%를 선택할까?
 - 서울 부동산 가격을 예로 들때 강남구만 테스트 세트로 뽑는게 서울 부동산 가격을 대표한다고 볼 수 있을까?

2. 테스트 세트 만들기

- 단순 랜덤하게 추출을 한다면, 전체 표본을 대표하는 샘플을 얻는 게 보장되지 않음 계층적 샘플링을 활용하여 표본 추출
- 도메인 지식을 활용하여, 표본 추출에 사용될 기준을 책정
 - 부동산 전문가에 따르면 해당 지역의 부동산가격은 해당 거주인의 소득과 크게 연관이 있다고함



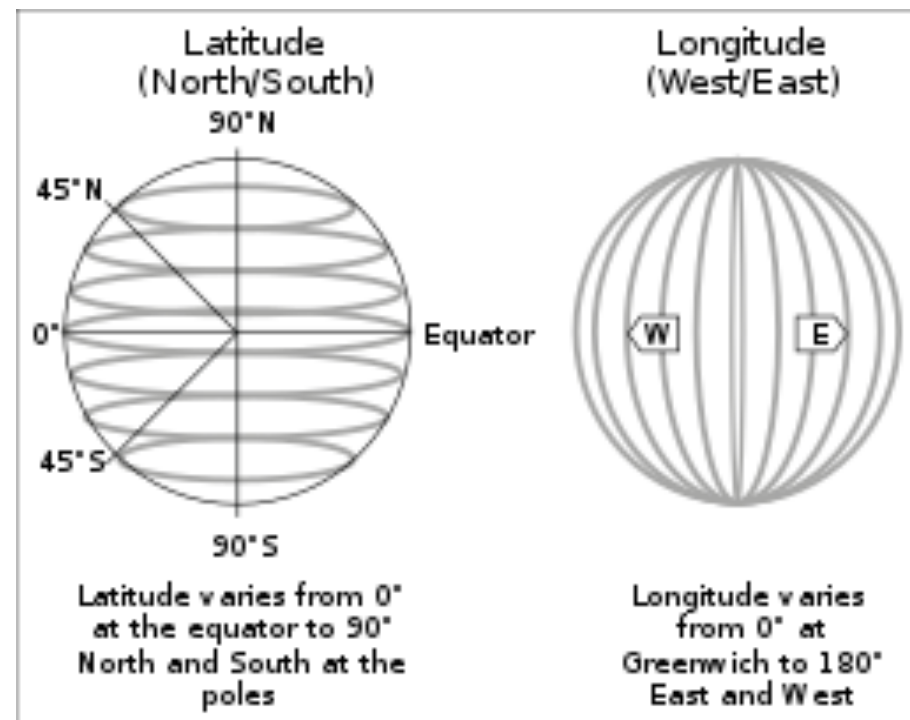
소득 카테고리의 히스토그램

	Overall	Stratified	Random	Rand. %error	Strat. %error
1	0.039826	0.039729	0.040213	0.973236	-0.243309
2	0.318847	0.318798	0.324370	1.732260	-0.015195
3	0.350581	0.350533	0.358527	2.266446	-0.013820
4	0.176308	0.176357	0.167393	-5.056334	0.027480
5	0.114438	0.114583	0.109496	-4.318374	0.127011

랜덤 샘플링 vs 계층 샘플링

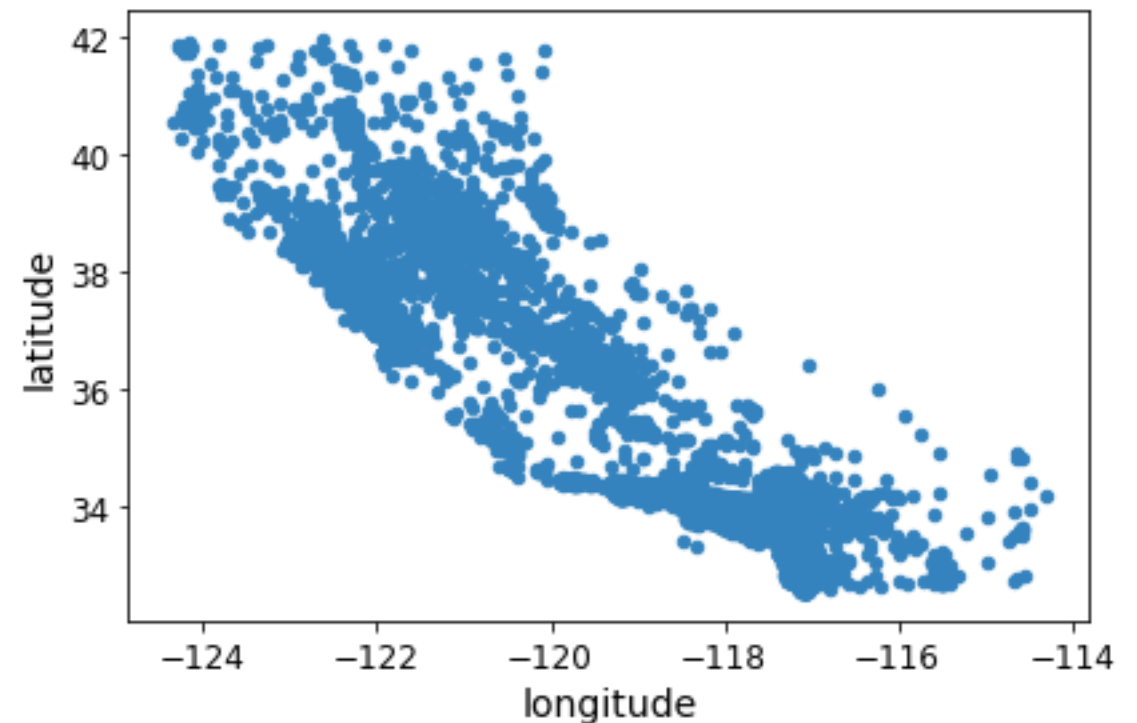
3. 탐색적 분석 (EDA)

- 경도(longitude), 위도(latitude)란?
 - 예) <https://www.google.co.kr/maps/place/Yonsei+University/@37.5657882,126.9363833,17z>
- 지리 좌표계



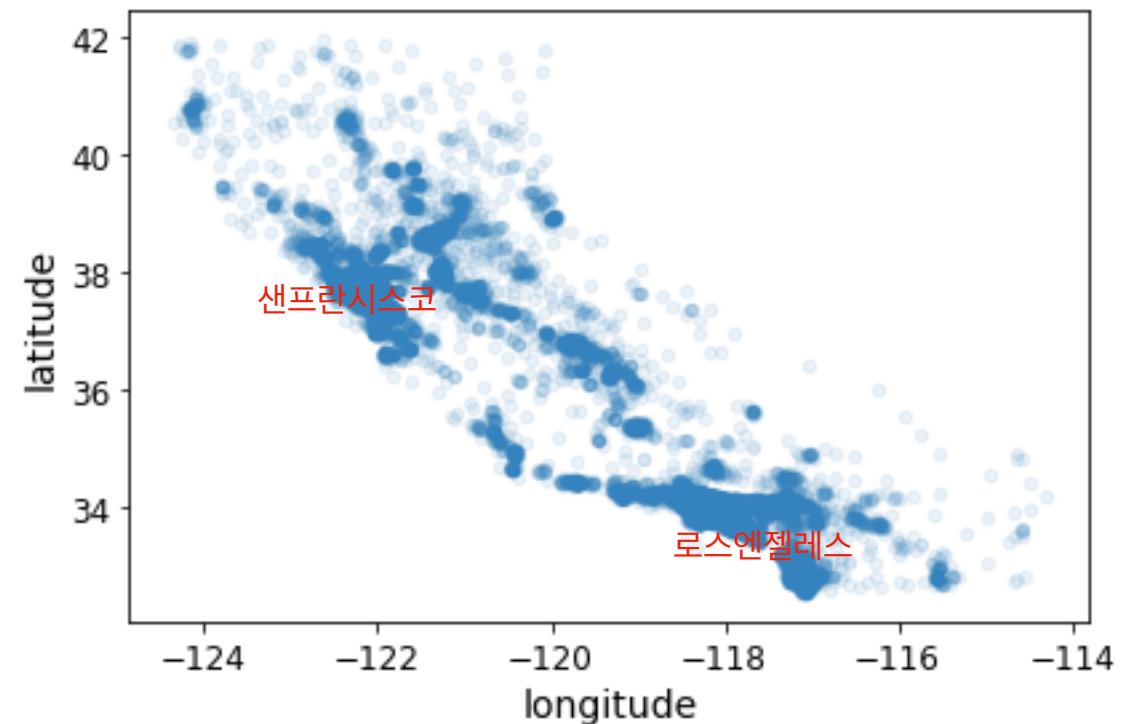
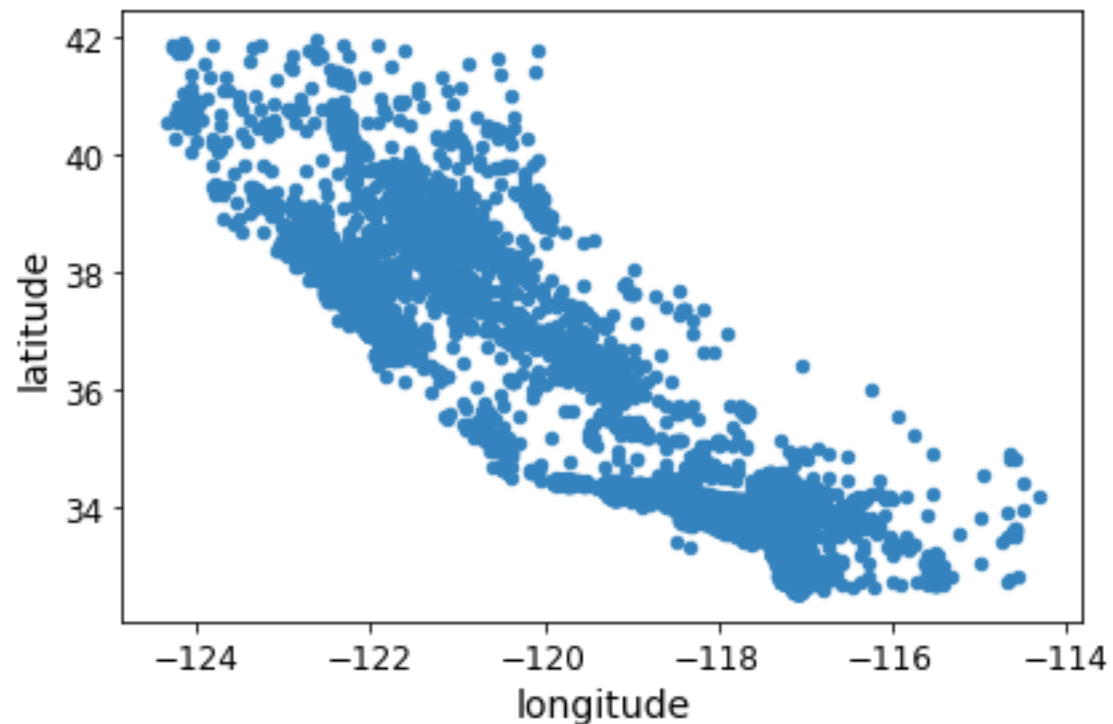
3. 탐색적 분석 (EDA)

- 데이터셋의 경도/위도를 이용하여 Scatter plot 출력
- 왼쪽 차트는 캘리포니아 지역을 잘 나타내지만, 특별한 인사이트를 얻을 수 없음



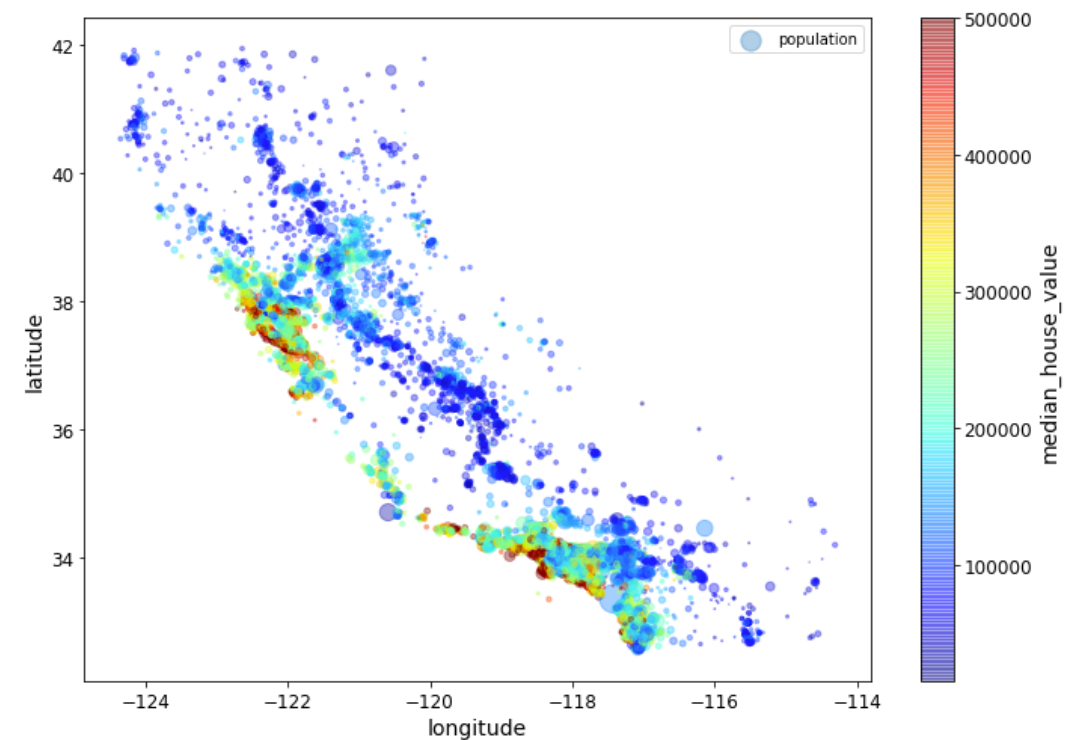
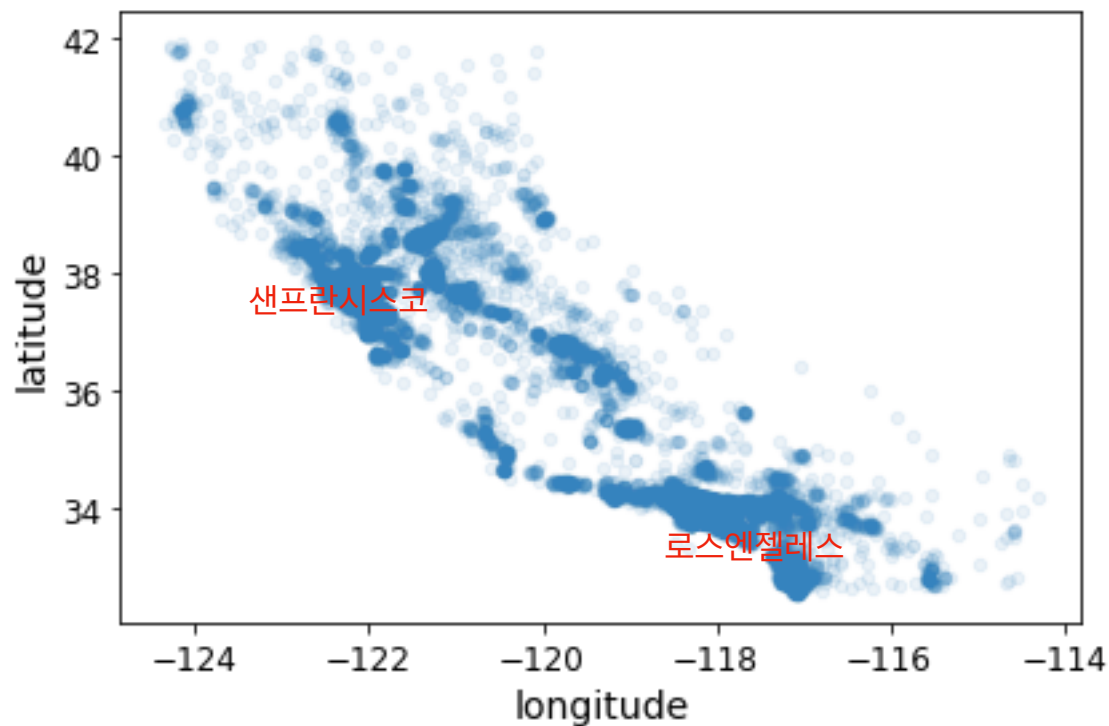
3. 탐색적 분석 (EDA)

- 각 데이터셋의 투명도를 조절하여 출력
- 데이터셋에서 밀집된 지역을 찾을 수 있음
 - 연한 부분은 사실 주 거주 지역이 아님 (교외)
 - 또 해안가를 따라 주로 밀집한다는 것을 볼 수 있음



3. 탐색적 분석 (EDA)

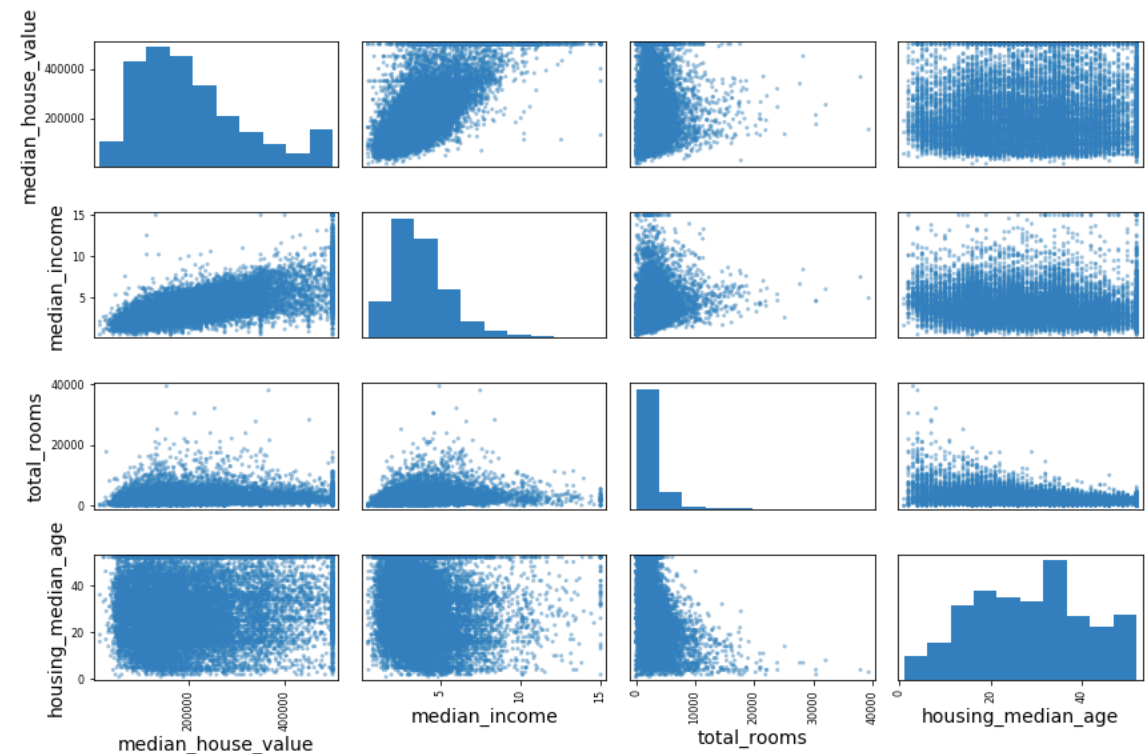
- 각 포인트에 색과 크기를 넣어 추가적인 속성을 표현함
 - 밀집된 지역의 중심가가 주로 높은 가격을 보임 (붉은색)
 - 외각 지역은 낮은 가격을 보임 (파란색)
 - 인구수가 많은 지역은 큰 동그라미 반대는 작은 동그라미
- 주택가격은 인구밀도와 연관이 있음



4. 상관관계 분석

- 우리가 예측하고자 하는 주택 가격과 가장 상관관계가 큰 속성 확인
 - 상관관계는 -1부터 1사이의 값으로 절대값을 취했을 때 1에 가까울 수록 두 속성이 높은 선형적인 상관관계를 보인다고 할 수 있음
 - 1: 음의 상관관계, 1: 양의 상관관계, 0: 상관관계 없음

median_house_value	1.000000
median_income	0.687160
total_rooms	0.135097
housing_median_age	0.114110
households	0.064506
total_bedrooms	0.047689
population	-0.026920
longitude	-0.047432
latitude	-0.142724



4. 상관관계 분석

- 속성들을 조합한 뒤에, 조합된 속성간의 상관관계 분석(feature 생성)
 - 예) 한 지역에 전체 방의 개수는 큰 의미가 없음, 중요한 것은 가구당 방 개수
- 조합의 예
 - rooms_per_household: 가구 당 방의 수 (total_rooms/households)
 - bedrooms_per_room: 방 중 침실의 비율 (total_bedrooms/total_rooms)
 - population_per_household: 가구당 거주자 수 (population/households)

median_house_value	1.000000
median_income	0.687160
total_rooms	0.135097
housing_median_age	0.114110
households	0.064506
total_bedrooms	0.047689
population	-0.026920
longitude	-0.047432
latitude	-0.142724

median_house_value	1.000000
median_income	0.687160
rooms_per_household	0.146285
total_rooms	0.135097
housing_median_age	0.114110
households	0.064506
total_bedrooms	0.047689
population_per_household	-0.021985
population	-0.026920
longitude	-0.047432
latitude	-0.142724
bedrooms_per_room	-0.259984

5. 데이터 전처리

- 결측값 처리

- total_bedrooms 특성값에 결측값이 존재
- 해당 행 (인스턴스) 삭제 (dropna)
- 해당 열 (속성) 삭제 (drop)
- 특정 값을 채우기 (fillna)

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 20640 entries, 0 to 20639  
Data columns (total 10 columns):  
longitude          20640 non-null float64  
latitude           20640 non-null float64  
housing_median_age 20640 non-null float64  
total_rooms         20640 non-null float64  
total_bedrooms      20433 non-null float64  
population          20640 non-null float64  
households          20640 non-null float64  
median_income       20640 non-null float64  
median_house_value  20640 non-null float64  
ocean_proximity     20640 non-null object  
dtypes: float64(9), object(1)  
memory usage: 1.6+ MB
```

5. 데이터 전처리

- 텍스트 또는 범주형 특성 다루기

- ocean_proximity는 범주형 데이터

```
housing["ocean_proximity"].value_counts()
```

```
<1H OCEAN    9136  
INLAND        6551  
NEAR OCEAN    2658  
NEAR BAY      2290  
ISLAND         5  
Name: ocean_proximity, dtype: int64
```

- Option1. Ordinal 형태로 변환

- <1H OCEAN : 0, INLAND: 1, NEAR OCEAN: 2, NEAR BAY: 3, ISLAND: 5
- good, normal, bad와 같이 위의 속성값들이 순서가 있다고 말 할 수 있나?

- Option2. One-hot 인코딩

- 속성 값들간의 우선순위가 보장되지 않는 경우에 보통 사용하는 방식
- <1H OCEAN : [1,0,0,0,0]
- INLAND : [0,1,0,0,0]
- NEAR OCEAN : [0,0,1,0,0]
- NEAR BAY : [0,0,0,1,0]
- ISLAND : [0,0,0,0,1]
- 1개였던 feature를 5개로 확장 각각은 이진속성

5. 데이터 전처리

- 속성 스케일링 (scaling)

- 각각의 숫자형 속성의 범위가 다를시 일부 모델(트리기반)을 제외하고는 학습성능이 떨어짐
- min-max scaling

$$x_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- 표준화 (standardization)

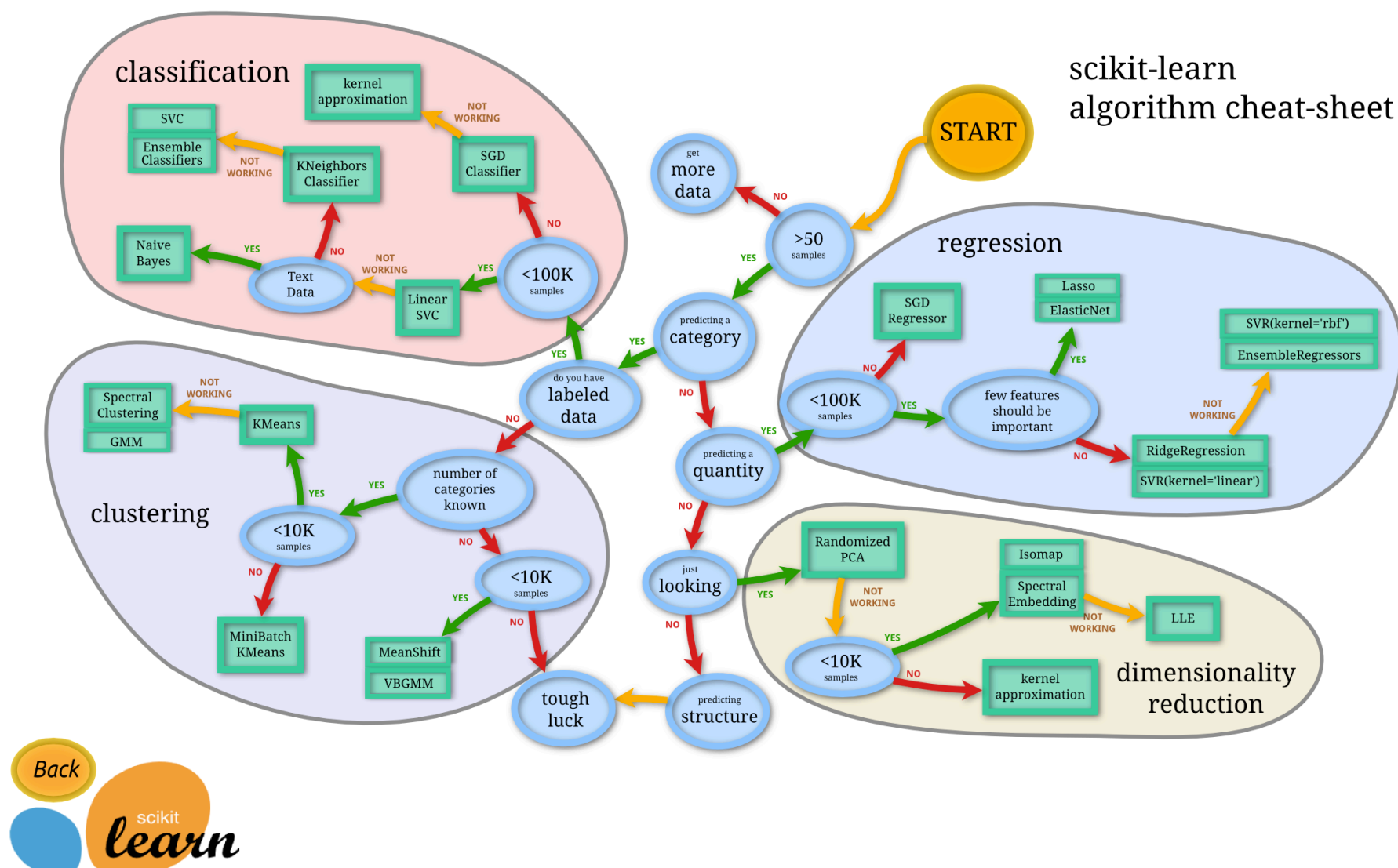
$$z = \frac{x - \mu}{\sigma}$$

```
housing.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

6. 모델 훈련 및 평가

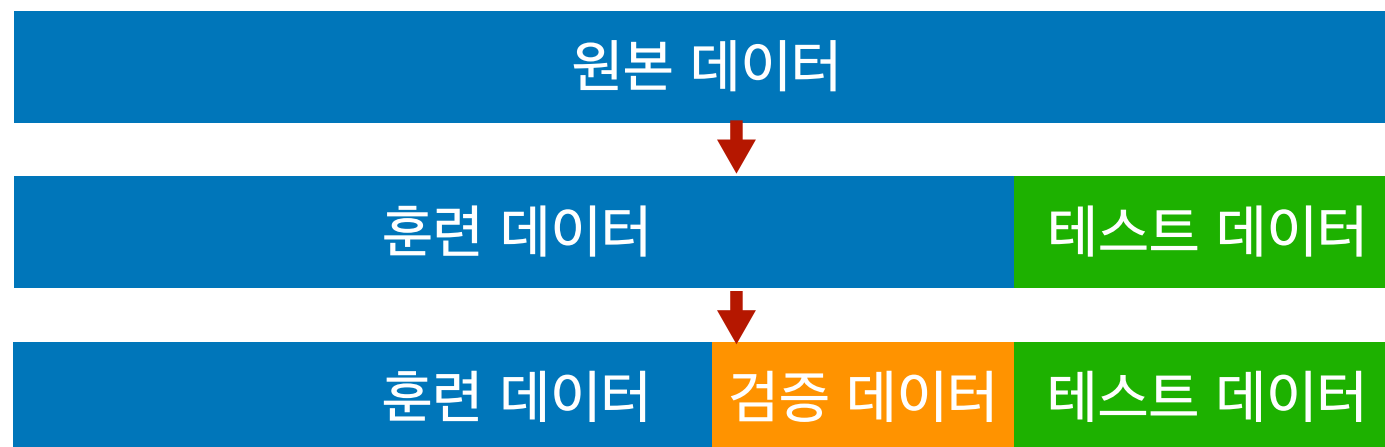
- 데이터 형태에 따라 암묵적으로 권장되는 모델이 있지만, 다양한 모델을 훈련하고 비교해보는 것이 일반적



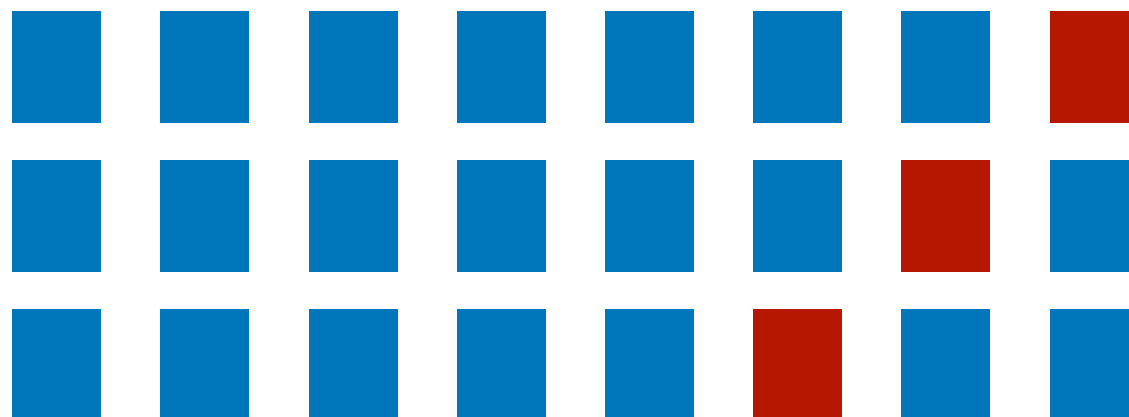
6. 모델 훈련 및 평가

- 교차 검증을 활용한 평가

- 원본 데이터 = 훈련 데이터 + 테스트 데이터
- 훈련 데이터 = 훈련 데이터 + 검증 데이터



- k-fold cross-validation
 - 훈련 데이터를 k 개로 나눈 뒤 훈련-검증데이터를 바꿔가며 모델의 성능평가



6. 모델 훈련 및 평가

- 모델 세부 튜닝

- 그리드 탐색 (grid search)를 이용한 최적의 parameter 탐색
 - 모든 파라미터의 조합을 섞어가며 최고의 조합을 찾음
 - 예) 의사 결정 tree의 max-depth, KNN의 k 값
- 랜덤 탐색을 (random search) 이용해 전체 조합을 테스트하는게 아니라 일부 임의의 조합을 테스트 해볼 수 있음
 - 그리드 서치에 비해 빠르지만 놓치는 조합이 있을 수 있음

- 앙상블 방법

- 여러 모델을 조합하여 하나의 모델을 형성
- 분류 문제의 경우 다수결로 결정
- 회귀 문제의 경우 평균값

- 모델을 완성 후 중요하지 않은 속성들을 파악한 뒤 제거

- feature importance

7. 테스트 세트로 시스템 평가

- 주의 사항

- 테스트 세트에 최적화되게 하이퍼파라미터를 수정하면 안됨!

참고문헌

- Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, 2019.

E.O.D