2장 머신러닝 프로젝트 처음부터 끝까지

감사의 글

자료를 공개한 저자 오렐리앙 제롱과 강의자료를 지원한 한빛아카데미에게 진심어린 감사를 전합니다.

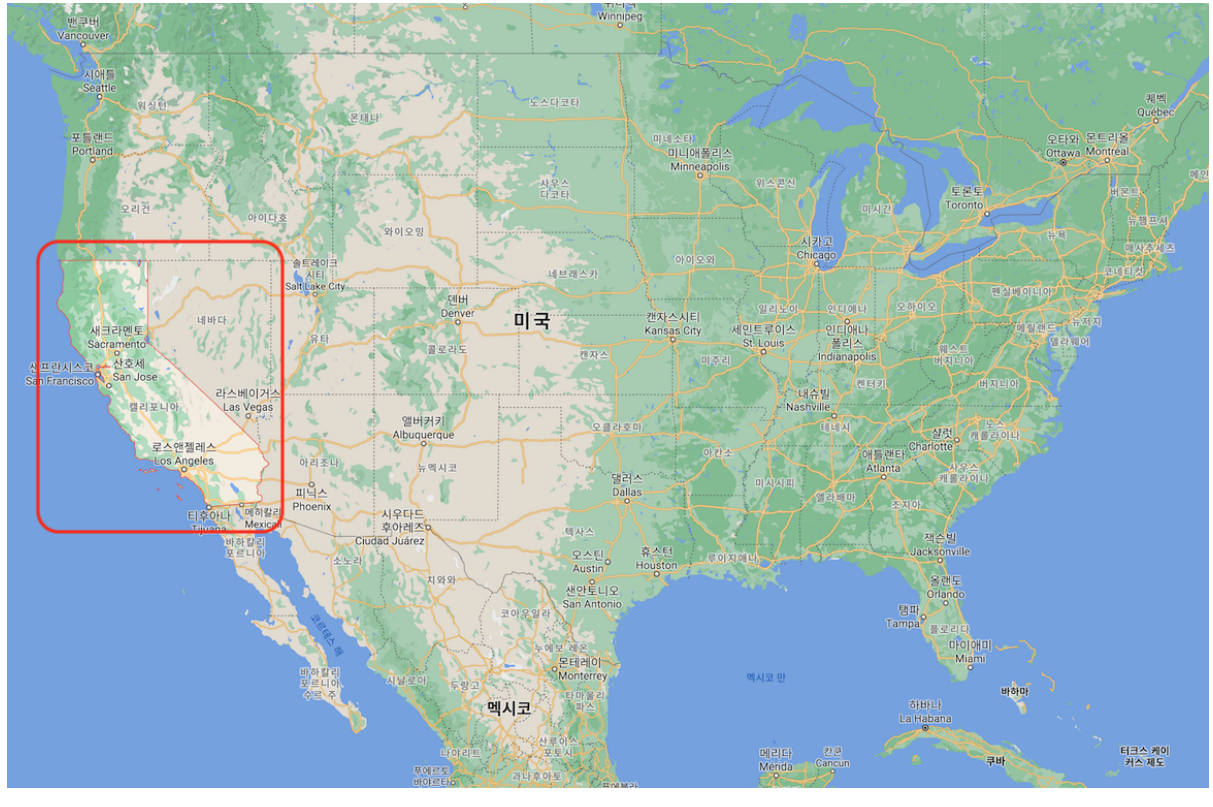
주요 내용

- 주택 가격을 예측하는 회귀 작업을 살펴보면서 선형 회귀, 결정 트리, 랜덤 포레스트 등 여러 알고리즘의 기본 사용법 소개
- 머신러닝 시스템 전체 훈련 과정 살펴보기

2.2 큰 그림 보기

주어진 데이터

- 미국 캘리포니아 주의 20,640개 지역별 인구조사 데이터
- 특성 10개: 경도, 위도, 중간 주택 연도, 방의 총 개수, 침실 총 개수, 인구, 가구 수, 중간 소득, 중간 주택 가격, 해안 근접도
- 목표: 구역별 중간 주택 가격 예측 시스템(모델) 구현하기



2.2.1 문제 정의

- 지도 학습(supervised learning)
    - 레이블: 구역별 중간 주택 가격
- 회귀(regression): 중간 주택 가격 예측
    - 다중 회귀(multiple regression): 여러 특성을 활용한 예측
    - 단변량 회귀(univariate regression): 구역마다 하나의 가격만 예측
- 배치 학습(batch learning): 빠르게 변하는 데이터에 적응할 필요가 없음

2.2.2 성능 측정 지표 선택

- 사용하는 모델에 따라 다른 모델 성능 측정 기준(norm) 선택
- 선형 회귀 모델의 경우 일반적으로 아래 두 기준 중 하나 사용
    - 평균 제곱근 오차(RMSE)
    - 평균 절대 오차(MAE)

평균 제곱근 오차(root mean square error, RMSE)

- 유클리디안 노름(Euclidean norm) 또는 ℓ_2 노름(norm)으로도 불림

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

- 기호 설명
    - $\mathbf{X}$: 훈련 데이터셋 전체 샘플들의 특성값들로 구성된 행렬, 레이블(타겟) 제외.
    - $\mathbf{x}^{(i)}$: i 번째 샘플의 전체 특성값 벡터. 레이블(타겟) 제외.
    - $y^{(i)}$: i 번째 샘플의 레이블
    - h: 예측 함수
    - $\hat{y}^{(i)} = h(\mathbf{x}^{(i)})$: i번째 샘플에 대한 예측 값

평균 절대 오차(mean absolute error, MAE)

- MAE는 맨해튼 노름 또는 ℓ_1 노름으로도 불림

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

- 이상치가 많은 경우 활용
- ℓ_1 노름과 ℓ_2 노름을 일반해서 ℓ_n 노름을 정의할 수도 있음
- RMSE가 MAE보다 이상치에 더 민감하지만, 이상치가 많지 않을 경우 일반적으로 RMSE 사용

2.3 데이터 가져오기

2.3.3 데이터 구조 훑어보기

첫 5개 데이터 살펴보기

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

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populatio
0	-122.23	37.88	41.0	880.0	129.0	322.(
1	-122.22	37.86	21.0	7099.0	1106.0	2401.(
2	-122.24	37.85	52.0	1467.0	190.0	496.(
3	-122.25	37.85	52.0	1274.0	235.0	558.(
4	-122.25	37.85	52.0	1627.0	280.0	565.(

전체 데이터셋 요약 정보

```
[6]   1 housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

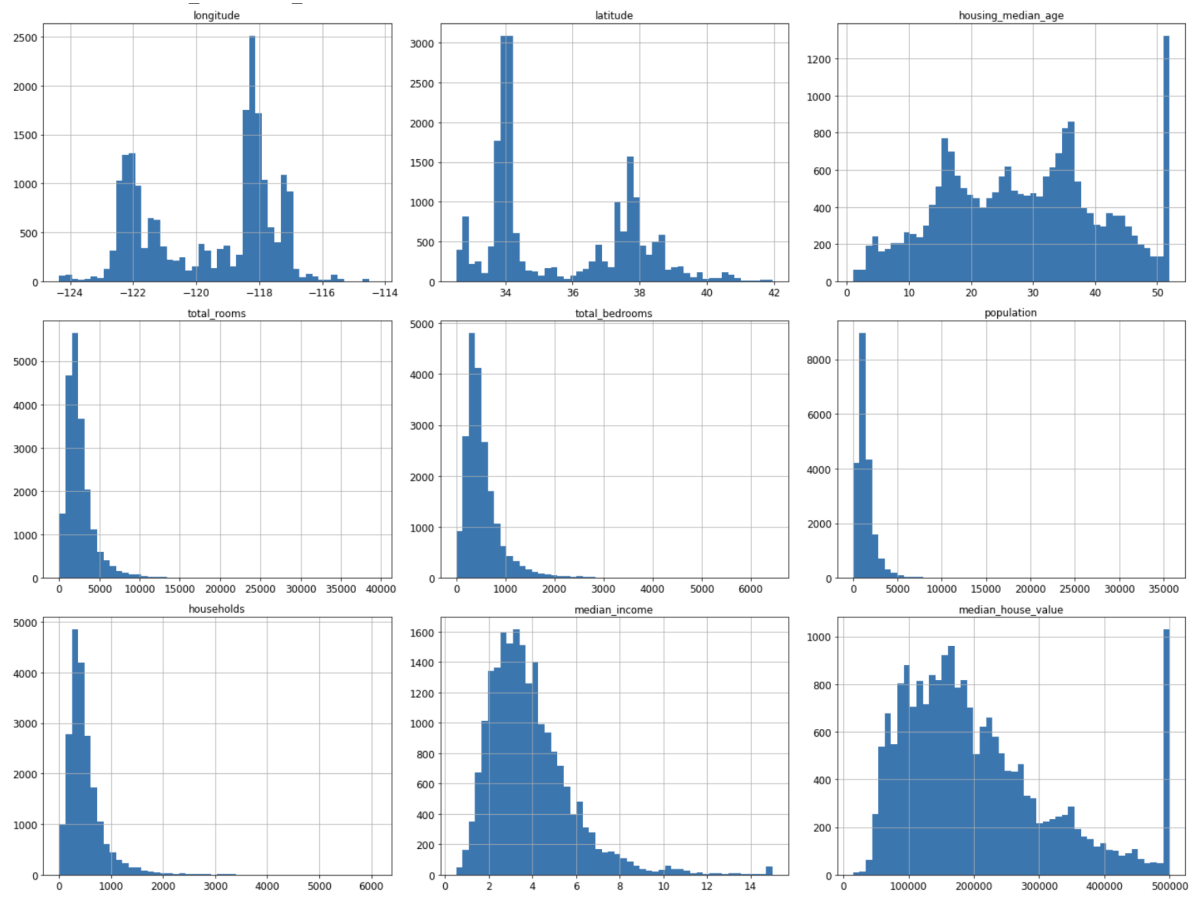
- 구역 수: 20,640개
- 구역별로 경도, 위도, 중간 주택 연도, 해안 근접도 등 총 10개의 조사 항목
    - '해안 근접도'는 범주형 특성이고 나머지는 수치형 특성.
- '방의 총 개수'의 경우 누락된 데이터인 207개의 null 값 존재

범주형 특성 탐색

- '해안 근접도'는 5개의 범주로 구분

특성값	설명
<1H OCEAN	해안에서 1시간 이내
INLAND	내륙
NEAR OCEAN	해안 근처
NEAR BAY	샌프란시스코의 Bay Area 지역
ISLAND	섬

수치형 특성별 히스토그램



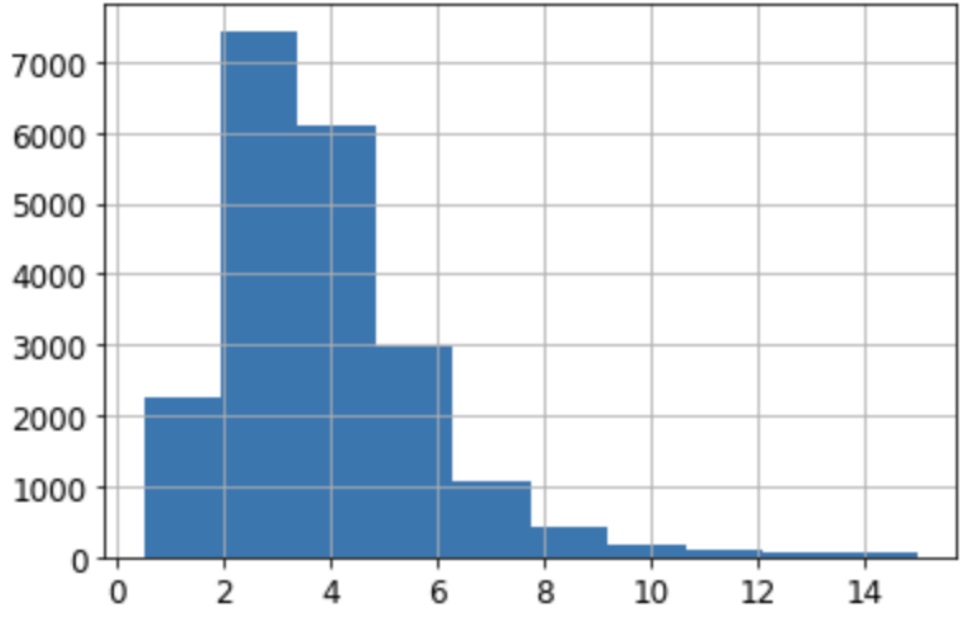
2.3.4 테스트 세트 만들기

- 모델 학습 시작 이전에 준비된 데이터셋을 훈련 세트과 테스트 세트로 구분
    - 테스트 세트 크기: 전체 데이터 셋의 20%
- 테스트 세트에 포함된 데이터는 미리 분석하지 말 것.
    - 미리 분석 시 **데이터 스누핑 편향**을 범할 가능성이 높아짐
    - 미리 보면서 알아낸 직관이 학습 모델 설정에 영향을 미칠 수 있음
- 훈련 세트와 데이터 세트를 구분하는 방식에 따라 결과가 조금씩 달라짐
    - 무작위 샘플링 vs. 계층적 샘플링
- 여기서는 계층적 샘플링 활용

계층적 샘플링

- 계층: 동질 그룹
    - 예제: 소득별 계층
- 테스트 세트: 전체 계층을 대표하도록 각 계층별로 적절한 샘플 추출
- 예제: 소득 범주
    - 계층별로 충분한 크기의 샘플이 포함되도록 지정해야 학습 과정에서 편향이 발생하지 않음
    - 특정 소득 구간에 포함된 샘플이 과하게 적거나 많으면 해당 계층의 중요도가 과대 혹은 과소 평가됨

- 전체 데이터셋의 중간 소득 히스토그램 활용



- 대부분 구역의 중간 소득이 **1.5~6.0**(15,000~60,000$) 사이
- 소득 구간을 아래 숫자를 기준으로 5개로 구분

```
[0, 1.5, 3.0, 4.6, 6.0, np,inf]
```

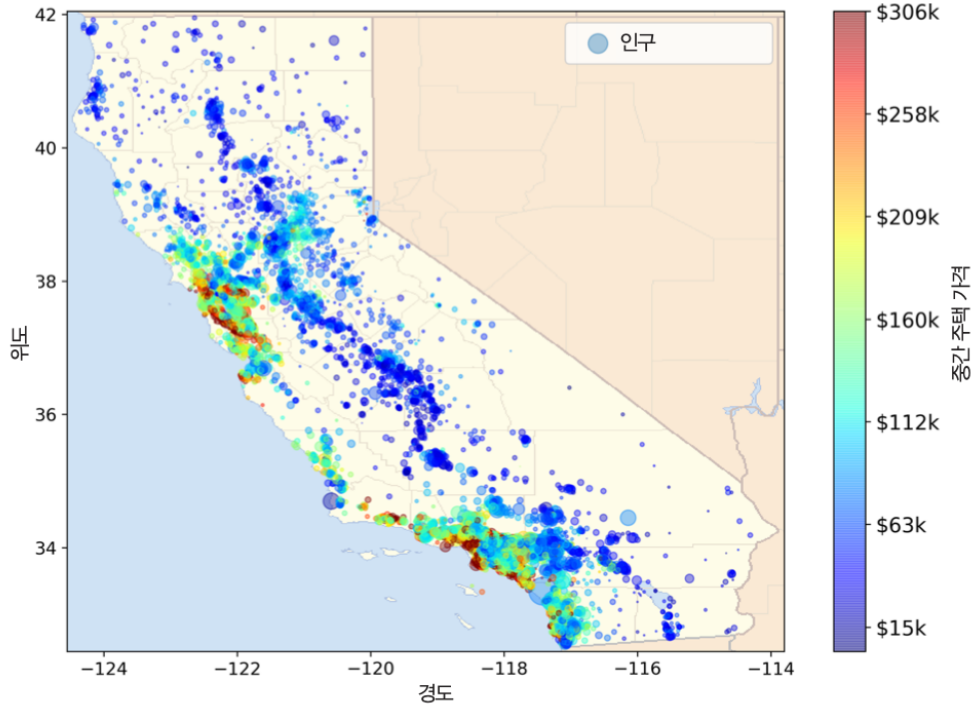
계층 샘플링과 무작위 샘플링 비교

	전체	계층 샘플링	무작위 샘플링	무작위 샘플링 오류율	계층 샘플링 오류율
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

2.4 데이터 이해를 위한 탐색과 시각화

2.4.1 지리적 데이터 시각화

- 주택 가격이 해안 근접도 또는 인구 밀도와 관련이 큼
- 해안 근접도: 위치에 따라 다르게 작용
    - 대도시 근처: 해안 근처 주택 가격이 상대적 높음
    - 북부 캘리포니아 지역: 높지 않음



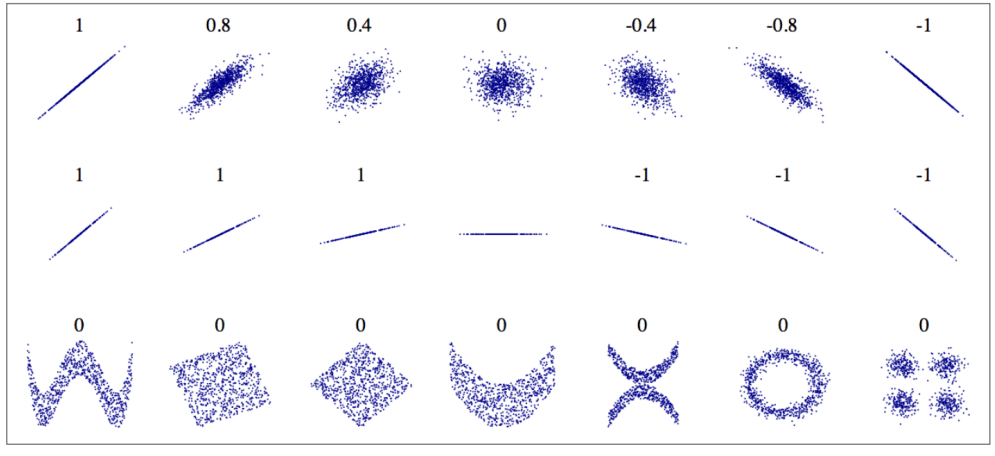
2.4.2 상관관계 조사

- 중간 주택 가격 특성과 다른 특성 사이의 상관관계: 상관계수 활용

```
In [39]: corr_matrix["median_house_value"].sort_values(ascending=False)
```

```
Out[39]: 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
         Name: median_house_value, dtype: float64
```

상관계수의 특징

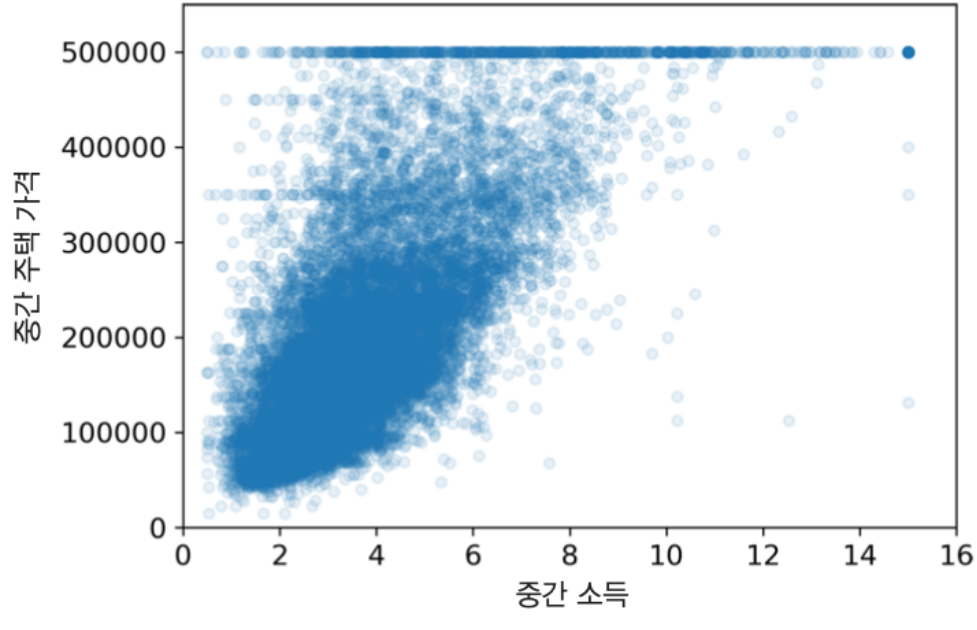


<그림 출처: 위키백과>

- 상관계수: $[-1, 1]$ 구간의 값
    - 1/-1에 가까울 수록: 강한 양/음의 선형 상관관계
    - 0에 가까울 수록: 매우 약한 선형 상관관계. 선형 관계가 아닌 다른 관계 존재 가능.
- 상관계수는 기울기와 아무 연관 없음

상관계수를 통해 확인할 수 있는 정보

- 중간 주택 가격과 중간 소득의 상관계수가 0.68로 가장 높음
    - 중간 소득이 올라가면 중간 주택 가격도 상승하는 경향이 있음
    - 점들이 너무 넓게 퍼져 있음. 완벽한 선형관계와 거리 멂.
- 50만, 45만, 35만, 28만 달러 수평선: 이유 불분명. 이상한 형태를 학습하지 않도록 해당 구역을 제거하는 것이 좋음.



2.4.3 특성 조합으로 실험

- 구역별 방의 총 개수와 침실의 총 개수 대신 아래 특성이 보다 유용함
    - 가구당 방 개수(rooms for household)
    - 방 하나당 침실 개수(bedrooms for room)
    - 가구당 인원(population per household)

```
[47]  1 corr_matrix = housing.corr()
      2 corr_matrix["median_house_value"].sort_values(ascending=False)
```

```
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
Name: median_house_value, dtype: float64
```

2.5 머신러닝 알고리즘을 위한 데이터 준비

데이터 전처리

- 데이터 전처리(data preprocessing): 효율적인 모델 훈련을 위한 데이터 변환
- 수치형 특성과 범주형 특성에 대해 다른 변환과정을 사용
- 수치형 특성 전처리 과정
    - 데이터 정제
    - 조합 특성 추가
    - 특성 스케일링
- 범주형 특성 전처리 과정
    - 원-핫-인코딩(one-hot-encoding)

2.5.1 데이터 정제: 수치형 특성 전치러 과정 1

- 누락된 특성값이 존재 경우, 해당 값 또는 특성을 먼저 처리해야 함.
- `total_bedrooms` 특성에 207개 구역에 대한 값이 null로 채워져 있음, 즉, 일부 구역에 대한 정보가 누락됨.

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_p
4629	-118.30	34.07	18.0	3759.0	NaN	3296.0	1462.0	2.2708	<1H
6068	-117.86	34.01	16.0	4632.0	NaN	3038.0	727.0	5.1762	<1H
17923	-121.97	37.35	30.0	1955.0	NaN	999.0	386.0	4.6328	<1H
13656	-117.30	34.05	6.0	2155.0	NaN	1039.0	391.0	1.6675	
19252	-122.79	38.48	7.0	6837.0	NaN	3468.0	1405.0	3.1662	<1H

null 값 처리 옵션

- 옵션 1: 해당 구역 제거
- 옵션 2: 전체 특성 삭제
- 옵션 3: 평균값, 중앙값, 0, 주변에 위치한 값 등 특정 값으로 채우기. 책에서는 중앙값으로 채움.

<옵션 3 활용>

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_p
4629	-118.30	34.07	18.0	3759.0	433.0	3296.0	1462.0	2.2708	<1H
6068	-117.86	34.01	16.0	4632.0	433.0	3038.0	727.0	5.1762	<1H
17923	-121.97	37.35	30.0	1955.0	433.0	999.0	386.0	4.6328	<1H
13656	-117.30	34.05	6.0	2155.0	433.0	1039.0	391.0	1.6675	
19252	-122.79	38.48	7.0	6837.0	433.0	3468.0	1405.0	3.1662	<1H

2.5.2 텍스트와 범주형 특성 다루기: 원-핫 인코딩

- 범주형 특성인 해안 근접도(ocean_proximity)를 수치형 특성으로 변환해야 함.
- 단순 수치화 적용 가능

범주	숫자
<1H OCEAN	0
INLAND	1
ISLAND	2
NEAR BAY	3
NEAR OCEAN	4

단순 수치화의 문제점

- 해안 근접도는 단순히 구분을 위해 사용. 해안에 근접하고 있다 해서 주택 가격이 기본적으로 더 비싸지 않음.
- 반면에 수치화된 값들은 크기를 비교할 수 있는 숫자
- 따라서 모델 학습 과정에서 숫자들의 크기 때문에 잘못된 학습이 이루어질 수 있음.

원-핫 인코딩(one-hot encoding)

- 수치화된 범주들 사이의 크기 비교를 피하기 위해 더미(dummy) 특성을 추가하여 활용
    - 범주 수 만큼의 더미 특성 추가
- 예를 들어, 해안 근접도 특성 대신에 다섯 개의 범주 전부를 새로운 특성으로 추가한 후 각각의 특성 값을 아래처럼 지정
    - 해당 카테고리의 특성값: 1
    - 나머지 카테고리의 특성값: 0

```
In [68]: cat_encoder = OneHotEncoder(sparse=False)
         housing_cat_1hot = cat_encoder.fit_transform(housing_cat)
         housing_cat_1hot

Out[68]: array([[1., 0., 0., 0., 0.],
                [1., 0., 0., 0., 0.],
                [0., 0., 0., 0., 1.],
                ...,
                [0., 1., 0., 0., 0.],
                [1., 0., 0., 0., 0.],
                [0., 0., 0., 1., 0.]])
```

2.5.4 특성 스케일링: 수치형 특성 전처리 과정 3

- 머신러닝 알고리즘은 입력 데이터셋의 특성값들의 스케일(범위)이 다르면 제대로 작동하지 않음
- 특성에 따라 다루는 숫자의 크기가 다를 때 통일된 스케일링이 필요
- 아래 두 가지 방식이 일반적으로 사용됨.
    - min-max 스케일링
    - 표준화 (책에서 사용)
- **주의**: 타깃(레이블)에 대한 스케일링은 하지 않음

min-max 스케일링

- **정규화**(normalization)라고도 불림
- 특성값 x를 $\frac{x-min}{max-min}$로 변환
- 변환 결과: **0에서 1** 사이
- 이상치에 매우 민감
    - 이상치가 매우 **크면 분모가 매우 커져서** 변환된 값이 **0 근처**에 몰림

표준화(standardization)

- 특성값 x 를 $\frac{x-\mu}{\sigma}$ 로 변환
    - μ: 특성값들의 **평균**값
    - σ: 특성값들의 **표준편차**
- 결과: 변환된 데이터들이 **표준정규분포**를 이룸
    - 이상치에 상대적으로 영향을 덜 받음.
- 사이킷런의 `StandardScaler` 변환기 활용 가능 (책에서 사용)

2.6 모델 선택과 훈련

- 목표 달성에 필요한 두 요소를 결정해야함
    - 학습 모델
    - 회귀 모델 성능 측정 지표
- 목표: 구역별 중간 주택 가격 예측 모델
- 학습 모델: 회귀 모델
- 회귀 모델 성능 측정 지표: 평균 제곱근 오차(RMSE)를 기본으로 사용

2.6.1 훈련 세트에서 훈련하고 평가하기

- 지금까지 한 일
    - 훈련 세트 / 테스트 세트 구분
    - 변환 파이프라인을 활용한 데이터 전처리
- 이제 할 일
    - 예측기 모델 선택 후 훈련시키기
    - 예제: 선형 회귀, 결정트리 회귀
- 예측기 모델 선택 후 훈련과정은 매우 단순함.
    - `fit()` 메서드를 전처리 처리가 된 훈련 데이터셋에 적용

선형 회귀 모델(4장)

- 선형 회귀 모델 생성: 사이킷런의 **`LinearRegression`** 클래스 활용
- 훈련 및 예측

```
from sklearn.linear_model import LinearRegression

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

  lin_reg.predict(housing_prepared)
```

선형 회귀 모델의 훈련 세트 대상 예측 성능

- RMSE(평균 제곱근 오차)가 68628.198 정도로 별로 좋지 않음.
- 훈련된 모델이 훈련 세트에 **과소적합** 됨.
    - 보다 좋은 특성을 찾거나 더 강력한 모델을 적용해야 함.
    - 보다 좋은 특성 예제: 로그 함수를 적용한 인구수 등

결정트리 회귀 모델(6장)

- 결정 트리 모델은 데이터에서 복잡한 비선형 관계를 학습할 때 사용
- 결정트리 회귀 모델 생성: 사이킷런의 `DecisionTreeRegressor` 클래스 활용
- 훈련 및 예측

```
from sklearn.tree import DecisionTreeRegressor

  tree_reg = DecisionTreeRegressor(random_state=42)
  tree_reg.fit(housing_prepared, housing_labels)

  housing_predictions = tree_reg.predict(housing_prepared)
```

결정트리 회귀 모델의 훈련 세트 대상 예측 성능

- RMSE(평균 제곱근 오차)가 0으로 완벽해 보임.
- 훈련된 모델이 훈련 세트에 심각하게 **과대적합** 됨.
    - 실전 상황에서 RMSE가 0이 되는 것은 불가능.
    - 훈련 세트가 아닌 테스트 세트에 적용할 경우 RMSE가 크게 나올 것임.

2.6.2 교차 검증을 사용한 평가

- 훈련 세트를 **폴드**(fold)라 불리는 k-개의 부분 집합으로 무작위로 분할
- 총 k 번 지정된 모델을 훈련: 훈련할 때마다 매번 다른 하나의 폴드 선택하여 검증 데이터셋으로 활용
- k = 5인 경우

- 결정 트리 모델 교차 검증(k = 10) RMSE: 약 71407 정도로 별로 좋지 않음.

2.7 모델 세부 튜닝

- 가능성이 높은 모델을 선정한 후에 **모델 세부 설정을 튜닝**해야함
- 튜닝을 위한 세 가지 방식
    - **그리드 탐색**
    - **랜덤 탐색**
    - **앙상블 방법**

2.7.1 그리드 탐색

- 지정한 하이퍼파라미터의 모든 조합을 교차검증하여 최선의 하이퍼파라미터 조합 찾기
- 사이킷런의 `GridSearchCV` 활용

예제: 그리드 탐색으로 랜덤 포레스트 모델에 대한 최적 조합 찾기

```
from sklearn.model_selection import GridSearchCV

param_grid = [
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
  ]

forest_reg = RandomForestRegressor(random_state=42)
grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)
grid_search.fit(housing_prepared, housing_labels)
```

- 총 (3x4 + 2x3 = 18) 가지의 경우 확인
- 5-겹 교차검증(`cv=5`)이므로, 총 (18x5 = 90)번 훈련함.

그리드 탐색 결과

- 최고 성능의 랜덤 포레스트 하이퍼파라미터가 다음과 같음.
    - `max_features` : 8
    - `n_estimators` : 30
    - 지정된 구간의 최고값들이기에 구간을 좀 더 넓히는 게 좋아 보임
- 최고 성능의 랜덤 포레스트에 대한 교차검증 RMSE: 49682

2.7.2 랜덤 탐색

- 그리드 탐색은 적은 수의 조합을 실험해볼 때 유용
- 조합의 수가 커지거나, 설정된 탐색 공간이 커지면 랜덤 탐색이 효율적
    - 설정값이 연속적인 값을 다루는 경우 랜덤 탐색이 유용
- 사이킷런의 `RandomizedSearchCV` 추정기가 랜덤 탐색을 지원

예제: 랜덤 탐색으로 랜덤 포레스트 모델에 대한 최적 조합 찾기

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

param_distribs = {
        'n_estimators': randint(low=1, high=200),
        'max_features': randint(low=1, high=8),
    }

forest_reg = RandomForestRegressor(random_state=42)
rnd_search = RandomizedSearchCV(forest_reg, param_distributions=param_distribs,
                                n_iter=10, cv=5,
scoring='neg_mean_squared_error', random_state=42)
rnd_search.fit(housing_prepared, housing_labels)
```

- `n_iter=10` : 랜덤 탐색이 총 10회 진행
    - `n_estimators`와 `max_features` 값을 지정된 구간에서 무작위 선택
- `cv=5` : 5-겹 교차검증. 따라서 랜덤 포레스트 학습이 (10x5=50)번 이루어짐.

랜덤 탐색 결과

- 최고 성능의 랜덤 포레스트 하이퍼파라미터가 다음과 같음.
    - `max_features`: 7
    - `n_estimators`: 180
- 최고 성능의 랜덤 포레스트에 대한 교차검증 RMSE: 49150

2.7.4 최상의 모델과 오차 분석

- 그리드 탐색과 랜덤 탐색 등을 통해 얻어진 최상의 모델을 분석해서 문제에 대한 좋은 통창을 얻을 수 있음
- 예를 들어, 최상의 랜덤 포레스트 모델에서 사용된 특성들의 중요도를 확인하여 일부 특성을 제외할 수 있음.
    - 중간 소득(median income)과 INLAND(내륙, 해안 근접도)가 가장 중요한 특성으로 확인됨
    - 해안 근접도의 다른 네 가지 특성은 별로 중요하지 않음
    - 중요도가 낮은 특성은 삭제할 수 있음.

```
[(0.36615898061813423, 'median_income'),
(0.16478099356159054, 'INLAND'),
(0.10879295677551575, 'pop_per_hhold'),
(0.07334423551601243, 'longitude'),
(0.06290907048262032, 'latitude'),
(0.056419179181954014, 'rooms_per_hhold'),
(0.053351077347675815, 'bedrooms_per_room'),
(0.04114379847872964, 'housing_median_age'),
(0.014874280890402769, 'population'),
(0.014672685420543239, 'total_rooms'),
(0.014257599323407808, 'households'),
(0.014106483453584104, 'total_bedrooms'),
(0.010311488326303788, '<1H OCEAN'),
(0.002856474637320158, 'NEAR OCEAN'),
(0.00196041559947807 06, 'NEAR BAY'),
(6.0280386727366e-05, 'ISLAND')]
```

최상의 모델 성능 배포