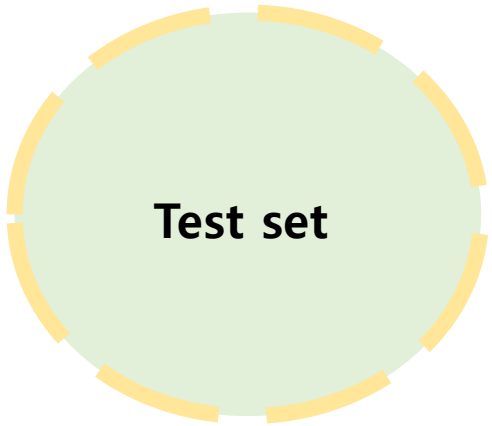
데이터 스누핑 편향



겉 : Y 패턴 → Y 패턴에 적합한 Y 머신러닝 모델 선정

속 : G 패턴

실제 성능 저하

"Test set 사전 검토하지 마세요"

In [27]:
```
np.random.seed(42)
```
값을 동일하게 설정하면 동일한 난수(random number)가 순서대로 생성

In [8]:
```
import numpy as np

def split_train_test(data, test_ratio):
    shuffled_indices = np.random.permutation(len(data))       # 배열을 섞음
    test_set_size = int(len(data) * test_ratio)               # 뒤섞인 정수에서 일정 비율만큼 test_set로 만들기, ratio 정하지 않
    test_indices = shuffled_indices[:test_set_size]           # 뒤섞인 정수에서 test_set 만큼 indexing
    train_indices = shuffled_indices[test_set_size:]          # 그 나머지를 train_set 로 indexing
    return data.iloc[train_indices], data.iloc[test_indices]

train_set, test_set = split_train_test(housing, 0.2)
len(train_set)
```

Out[8]:
```
16512
```

In [32]:
```
len(test_set)
```

Out[32]:
```
4128
```

In [39]:
```
# 데이터 업데이트할 경우, test -> train 으로 넘어가지 않도록

from zlib import crc32

def test_set_check(identifier, test_ratio):
    return crc32(np.int64(identifier)) & 0xffffff < test_ratio * 2**32
```
test_ratio보다 큰 양의 정수는 모두 training으로 사용

In [40]:
```
def split_train_test_by_id(data, test_ratio, id_column):
    ids = data[id_column]
    in_test_set = ids.apply(lambda id_:test_set_check(id_, test_ratio))
    return data.loc[~in_test_set], data.loc[in_test_set]
```

In [41]:
```
# index 열이 추가된 데이터 프레임 반환
housing_with_id = housing.reset_index()

train_set, test_set = split_train_test_by_id(housing_with_id, 0.2, "index")
```

1) np.random.seed( )

- 각 노트북의 실행 결과가 동일
- 하지 않으면? : 프로그램 재실행 시, 다른 test set 생성

→ML 알고리즘이 전체 데이터 보게 됨

2) Id를 하나씩 뽑아 crc32에서 정의한 범위에 적합한지 check

- Test_ratio 보다 작으면 True → test set
- Test_ratio 보다 크면 False → train set

3) Housing data sets에는 식별자 없으므로 index로 대신 함

→ 하나의 행도 삭제 되면 안됨 (??)

4) Id = 캘리포니아 주택 구역의 경도+위도 (특성을 조합해 새로운 ID 만든 경우, ID가 정말 고유한지 확인)

In [43]:
```
housing_with_id["id"] = housing["longitude"] * 1000 + housing["latitude"]
train_set, test_set = split_train_test_by_id(housing_with_id, 0.2, "id")
```

In [45]:
```
test_set.head(10)
```

Out[45]:

ngitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity	id
-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY	-122192.12
-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY	-122182.14
-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY	-122202.15
-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY	-122212.15
-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY	-122212.15
-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	269700.0	NEAR BAY	-122212.15
-122.25	37.84	52.0	2535.0	489.0	1094.0	514.0	3.6591	299200.0	NEAR BAY	-122212.16
-122.25	37.84	52.0	3104.0	687.0	1157.0	647.0	3.1200	241400.0	NEAR BAY	-122212.16
-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	226700.0	NEAR BAY	-122222.16
-122.25	37.84	52.0	3549.0	707.0	1551.0	714.0	3.6912	261100.0	NEAR BAY	-122212.16

5) split_train_test → train_test_split

기본적으로 Shuffled 기능 포함된 함수

In [21]:
```
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)

test_set.head()
```

Out[21]:

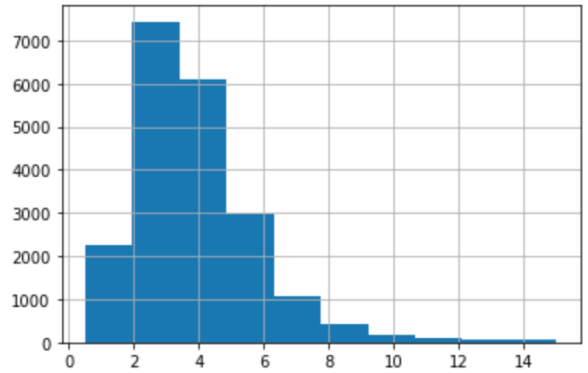
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
20046	-119.01	36.06	25.0	1505.0	NaN	1392.0	359.0	1.6812	47700.0	INLAND
3024	-119.46	35.14	30.0	2943.0	NaN	1565.0	584.0	2.5313	45800.0	INLAND
15663	-122.44	37.80	52.0	3830.0	NaN	1310.0	963.0	3.4801	500001.0	NEAR BAY
20484	-118.72	34.28	17.0	3051.0	NaN	1705.0	495.0	5.7376	218600.0	<1H OCEAN
9814	-121.93	36.62	34.0	2351.0	NaN	1063.0	428.0	3.7250	278000.0	NEAR OCEAN

설문조사 : 1000명/78억명 (78억명 중 여성=51.3%, 남성 48.7%)

1) 무작위 샘플링 → 남녀 비율이 어떻게 될지 모름 ex. 여성 200명, 남성 800명 → 결과 편향

2) 계층적 샘플링 → 여성 513명, 남성 487명

```
In [16]: housing["median_income"].hist()
Out[16]: <AxesSubplot:>
```



6) 연속데이터인 "median_income" → 카테고리 특성을 가진 범주형으로 변환

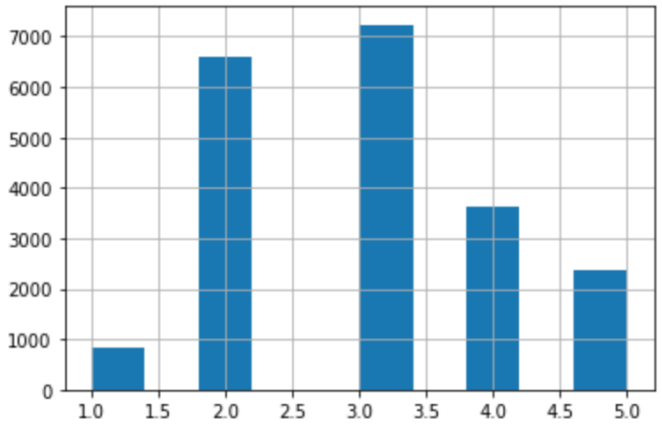
7) pd.cut : 구역 분할

```
In [13]: housing["income_cat"] = pd.cut(housing["median_income"],
                                        bins=[0., 1.5, 3.0, 4.5, 6., np.inf],  ← 5개 구역으로 분할
                                        labels=[1, 2, 3, 4, 5])

In [14]: housing["income_cat"].value_counts()

Out[14]: 3    7236
         2    6581
         4    3639
         5    2362
         1     822
         Name: income_cat, dtype: int64
```

```
In [15]: housing["income_cat"].hist()
Out[15]: <AxesSubplot:>
```



8) 소득 카테고리(범주형 특성) → 계층 샘플링

In [17]:
```
from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(housing, housing["income_cat"]):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]
```

n_split=1 : test, train 2개로 분할

Labels → test / train 균등 배분

9) 계층 샘플링한 테스트 세트에서 소득 카테고리의 비율

In [18]:
```
strat_test_set["income_cat"].value_counts() / len(strat_test_set)
```

Out[18]:
```
3    0.350533
2    0.318798
4    0.176357
5    0.114583
1    0.039729
Name: income_cat, dtype: float64
```

"test_set" = 전체 data sets에 있는 여러 소득 카테고리 대표

10) train_test_split (무작위 샘플링) vs stratified train_test_split (계층 샘플링)

In [24]:
```
def income_cat_proportions(data):
    return data["income_cat"].value_counts() / len(data)

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

compare_props = pd.DataFrame({
    "Overall" : income_cat_proportions(housing),
    "Stratified" : income_cat_proportions(strat_test_set),
    "Random" : income_cat_proportions(test_set),
}).sort_index()
compare_props["Rand. %error"] = 100 * compare_props["Random"] / compare_props["Overall"] - 100
compare_props["Strat. %error"] = 100 * compare_props["Stratified"] / compare_props["Overall"] - 100
```

In [25]:
```
compare_props
```

Out[25]:

	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

11) 데이터 원상 복귀 (소득 카테고리 특성 삭제)

In [26]:
```
for set_in (st) (strat_train_set, strat_test_set):
    set_.drop("income_cat"), axis=1, inplace=True)
```