In [1]:
```
# 機械学習　p.64～73

# 2.5 機械学習アルゴリズムが処理しやすいようにデータを準備する
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import StratifiedShuffleSplit

housing = pd.read_csv("datasets/housing/housing.csv")

housing["income_cat"] = pd.cut(housing["median_income"],
                               bins=[0.0, 1.5, 3.0, 4.5, 6.0,np.inf], labels=[1, 2, 3, 4, 5])
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]

for set_ in (strat_train_set, strat_test_set):
    set_.drop("income_cat", axis=1, inplace=True)

housing = strat_train_set.drop("median_house_value", axis=1)
housing_labels = strat_train_set["median_house_value"].copy()
```

In [3]:
```
# longitude latitude housing_median_age total_rooms total_bedrooms households median_income median_house_value ocean_proximity
# 経度      緯度     築年数の中央値      部屋数       寝室数          世帯数     収入の中央値   住宅価格の中央値     海との位置関係
```

In [4]:
```
# scikit-learn の SimpleImputer による欠損値の埋め込み

housing_num = housing.drop("ocean_proximity", axis=1) # テキスト属性の除去

from sklearn.impute import SimpleImputer # クラスのインポート

imputer = SimpleImputer(strategy="median") # 「中央値埋め込む」インスタンスの作成

imputer.fit(housing_num) # データにインスタンスを適合
x = imputer.transform(housing_num) # 欠損値の埋め込み
housing_num = pd.DataFrame(x, columns=housing_num.columns, index=housing_num.index) # DataFrame に戻す
```

In [5]:
```
# 埋め込み実験（テストデータ）
test = pd.read_csv("datasets/test/test.csv")
test
```

Out[5]:

	term-A	term-B	term-C	term-D	term-E
0	1	10	15.0	100	1000
1	2	20	NaN	200	2000
2	3	30	NaN	300	3000
3	4	40	NaN	400	4000
4	5	50	55.0	500	5000

In [6]:
```
imputer.fit(test)
temp = imputer.transform(test)
test = pd.DataFrame(temp, columns=test.columns, index=test.index)
test
```

Out[6]:

	term-A	term-B	term-C	term-D	term-E
0	1.0	10.0	15.0	100.0	1000.0
1	2.0	20.0	35.0	200.0	2000.0
2	3.0	30.0	35.0	300.0	3000.0
3	4.0	40.0	35.0	400.0	4000.0
4	5.0	50.0	55.0	500.0	5000.0

In [7]:
```
# テキスト／カテゴリ属性の処理
housing_cat = housing[["ocean_proximity"]]
housing_cat.head(10)
```

Out[7]:

	ocean_proximity
17606	<1H OCEAN
18632	<1H OCEAN
14650	NEAR OCEAN

	ocean_proximity
3230	INLAND
3555	<1H OCEAN
19480	INLAND
8879	<1H OCEAN
13685	INLAND
4937	<1H OCEAN
4861	<1H OCEAN

In [8]:
```
# OrdinalEncoder

from sklearn.preprocessing import OrdinalEncoder

ordinal_encoder = OrdinalEncoder()
housing_cat_encoded = ordinal_encoder.fit_transform(housing_cat)
housing_cat_encoded[:10]
```

Out[8]:
```
array([[0.],
       [0.],
       [4.],
       [1.],
       [0.],
       [1.],
       [0.],
       [1.],
       [0.],
       [0.]])
```

In [9]:
```
ordinal_encoder.categories_
```

Out[9]:
```
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
       dtype=object)]
```

In [10]:
```
# OneHotEncoder

from sklearn.preprocessing import OneHotEncoder

cat_encoder = OneHotEncoder()
housing_cat_1hot = cat_encoder.fit_transform(housing_cat)
housing_cat_1hot
```

Out[10]:
```
<16512x5 sparse matrix of type '<class 'numpy.float64'>'
        with 16512 stored elements in Compressed Sparse Row format>
```

In [11]:
```
housing_cat_1hot.toarray()
```

Out[11]:
```
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.]])
```

In [12]:
```
# カスタム変換器 BaseEstimator, TransformerMixin

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): # *args, **kargs なし
        self.add_bedrooms_per_room = add_bedrooms_per_room

    def fit(self, X, y=None):
        return self # 特になし

    def transform(self, X):
        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_house_hold, population_per_household]
```

In [13]:
```
# 特徴量のスケーリング MinMaxScaler:最小最大スケーリング、StandardScaler:標準化
```

In [14]:
```
# 変換パイプライン

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

num_pipeline = Pipeline([
```

```
    ('imputer', SimpleImputer(strategy="median")),
    ('attribs_adder', CombinedAttributesAdder()),
    ('std_scaler', StandardScaler()),
])

housing_num_tr = num_pipeline.fit_transform(housing_num)
housing_num_tr
```

Out[14]:
```
array([[-1.15604281,  0.77194962,  0.74333089, ..., -0.31205452,
        -0.08649871,  0.15531753],
       [-1.17602483,  0.6596948 , -1.1653172 , ...,  0.21768338,
        -0.03353391, -0.83628902],
       [ 1.18684903, -1.34218285,  0.18664186, ..., -0.46531516,
        -0.09240499,  0.4222004 ],
       ...,
       [ 1.58648943, -0.72478134, -1.56295222, ...,  0.3469342 ,
        -0.03055414, -0.52177644],
       [ 0.78221312, -0.85106801,  0.18664186, ...,  0.02499488,
         0.06150916, -0.30340741],
       [-1.43579109,  0.99645926,  1.85670895, ..., -0.22852947,
        -0.09586294,  0.10180567]])
```

In [15]:
```
# ColumnTransformer

from sklearn.compose import ColumnTransformer

num_attribs = list(housing_num)
cat_attribs = ["ocean_proximity"]

full_pipeline = ColumnTransformer([
    ("num", num_pipeline, num_attribs),
    ("cat", OneHotEncoder(), cat_attribs),
])

housing_prepared = full_pipeline.fit_transform(housing)
housing_prepared
```

Out[15]:
```
array([[-1.15604281,  0.77194962,  0.74333089, ...,  0.        ,
         0.        ,  0.        ],
       [-1.17602483,  0.6596948 , -1.1653172 , ...,  0.        ,
         0.        ,  0.        ],
       [ 1.18684903, -1.34218285,  0.18664186, ...,  0.        ,
         0.        ,  1.        ],
       ...,
       [ 1.58648943, -0.72478134, -1.56295222, ...,  0.        ,
         0.        ,  0.        ],
       [ 0.78221312, -0.85106801,  0.18664186, ...,  0.        ,
         0.        ,  0.        ],
```

```
       [-1.43579109,  0.99645926,  1.85670895, ...,  0.        ,
         1.        ,  0.        ]])
```