In [1]:
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import  StratifiedShuffleSplit
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import SplineTransformer,OneHotEncoder
from sklearn.preprocessing import StandardScaler


#1. Load the dataset
housing = pd.read_csv("housing.csv")

#2. Create a stratified test set
housing["income_cat"] = pd.cut(housing['median_income'],
                               bins=[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].drop("income_cat",axis=1)
    strat_test_set = housing.loc[test_index].drop("income_cat",axis=1)

# We  will work on the copy of training data
housing = strat_train_set.copy()

#3. Seprate fetures and labels
housing_labels = housing["median_house_value"].copy()
housing =housing.drop("median_house_value",axis=1)

print(housing,housing_labels)


#4. List and Seprate numerical and categorical columns
num_attribs = housing.drop("ocean_proximity",axis=1).columns.tolist()
cat_attribs = ["ocean_proximity"]

#5. Lets make the pipeline

# For numerical columns
num_pipeline = Pipeline([
    ("impute",SimpleImputer(strategy="median")),
    ("Standardize",StandardScaler())
    ])


# For Categoriacl columns
cat_pipeline = Pipeline([
    ("onehot",OneHotEncoder(handle_unknown="ignore"))
    ])

# Construct the full pipeline
full_pipeline = ColumnTransformer([
    ("num",num_pipeline,num_attribs),
    ("cat",cat_pipeline,cat_attribs)
])
```

```python
#6. Transform the data
housing_prepared = full_pipeline.fit_transform(housing)
print(housing_prepared)
```

```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
12655    -121.46     38.52                  29         3873           797.0
15502    -117.23     33.09                   7         5320           855.0
2908     -119.04     35.37                  44         1618           310.0
14053    -117.13     32.75                  24         1877           519.0
20496    -118.70     34.28                  27         3536           646.0
...          ...       ...                 ...          ...             ...
15174    -117.07     33.03                  14         6665          1231.0
12661    -121.42     38.51                  15         7901          1422.0
19263    -122.72     38.44                  48          707           166.0
19140    -122.70     38.31                  14         3155           580.0
19773    -122.14     39.97                  27         1079           222.0

       population  households  median_income ocean_proximity
12655        2237         706         2.1736          INLAND
15502        2015         768         6.3373      NEAR OCEAN
2908          667         300         2.8750          INLAND
14053         898         483         2.2264      NEAR OCEAN
20496        1837         580         4.4964       <1H OCEAN
...           ...         ...            ...             ...
15174        2026        1001         5.0900       <1H OCEAN
12661        4769        1418         2.8139          INLAND
19263         458         172         3.1797       <1H OCEAN
19140        1208         501         4.1964       <1H OCEAN
19773         625         197         3.1319          INLAND

[16512 rows x 9 columns] 12655     72100
15502    279600
2908      82700
14053    112500
20496    238300
          ...
15174    268500
12661     90400
19263    140400
19140    258100
19773     62700
Name: median_house_value, Length: 16512, dtype: int64
[[-0.94135046  1.34743822  0.02756357 ...  0.          0.
   0.        ]
 [ 1.17178212 -1.19243966 -1.72201763 ...  0.          0.
   1.        ]
 [ 0.26758118 -0.1259716   1.22045984 ...  0.          0.
   0.        ]
 ...
 [-1.5707942   1.31001828  1.53856552 ...  0.          0.
   0.        ]
 [-1.56080303  1.2492109  -1.1653327  ...  0.          0.
   0.        ]
 [-1.28105026  2.02567448 -0.13148926 ...  0.          0.
   0.        ]]
```

In [2]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import root_mean_squared_error
```

7. Train the model

Linear Regression

In [3]:
```
# lin_reg = LinearRegression()
# lin_reg.fit(housing_prepared,housing_labels)
# lin_preds = lin_reg.predict(housing_prepared)
# # lin_rmse = root_mean_squared_error(housing_labels,lin_preds)
```

In [4]:
```
# print(f"The root mean squared error for Linear Regression is {lin_rmse}")
```

Decision Tree

In [5]:
```
#dec_reg = DecisionTreeRegressor()
#dec_reg.fit(housing_prepared,housing_labels)
#dec_preds = dec_reg.predict(housing_prepared)
#dec_rmse = root_mean_squared_error(housing_labels,dec_preds)
```

In [6]:
```
# print(f"The root mean squared error for Desision tree is {dec_rmse}")
```

Random Forest Model

In [7]:
```
# random_forest_reg = RandomForestRegressor()
# random_forest_reg.fit(housing_prepared,housing_labels)
# random_forest_preds = random_forest_reg.predict(housing_prepared)
# random_forest_rmse = root_mean_squared_error(housing_labels,random_forest_pred
```

In [8]:
```
# print(f"The root mean squared error for Random Forest is {random_forest_rmse}"
```

8. Cross Validattion

In [9]:
```
from sklearn.model_selection import cross_val_score
```

In [10]:
```
dec_reg = DecisionTreeRegressor()
dec_reg.fit(housing_prepared,housing_labels)
dec_preds = dec_reg.predict(housing_prepared)
#dec_rmse = root_mean_squared_error(housing_labels,dec_preds)
dec_rmses = -cross_val_score(dec_reg,housing_prepared,housing_labels,scoring="ne

print(pd.Series(dec_rmses).describe())

random_forest_reg = RandomForestRegressor()
random_forest_reg.fit(housing_prepared,housing_labels)
random_forest_preds = random_forest_reg.predict(housing_prepared)
# random_forest_rmse = root_mean_squared_error(housing_labels,random_forest_pred
random_forest_rmses = -cross_val_score(random_forest_reg,housing_prepared,housin

print(pd.Series(random_forest_rmses).describe())
```

```
lin_reg = LinearRegression()
lin_reg.fit(housing_prepared,housing_labels)
lin_preds = lin_reg.predict(housing_prepared)
#lin_rmse = root_mean_squared_error(housing_labels,lin_preds)
lin_rmses = -cross_val_score(lin_reg,housing_prepared,housing_labels,scoring="ne

print(pd.Series(lin_rmses).describe())
```

```
count       10.000000
mean     69329.806956
std       2717.228989
min      63732.461319
25%      68030.605234
50%      69593.605180
75%      71136.802180
max      73163.033739
dtype: float64
count       10.000000
mean     49370.008603
std       2089.715241
min      46048.804428
25%      47851.130605
50%      49023.234196
75%      50631.029659
max      52990.201716
dtype: float64
count       10.000000
mean     69204.322755
std       2500.382157
min      65318.224029
25%      67124.346106
50%      69404.658178
75%      70697.800632
max      73003.752739
dtype: float64
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: