In [6]:
```
import os
import joblib
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
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import root_mean_squared_error
from sklearn.model_selection import cross_val_score
```

In [7]:
```
MODEL_FILE = "model.pkl"
PIPELINE_FILE = "pipeline.pkl"
```

In [10]:
```
def build_pipeline(num_attribs,cat_attribs):
#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)
    ])

    return full_pipeline
```

In [11]:
```
if not os.path.exists(MODEL_FILE):

     #Lets train the model
    housing = pd.read_csv("housing.csv")
    #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"]):
        housing.loc[test_index].drop("income_cat",axis=1).to_csv("input.csv",ind
        housing = housing.loc[train_index].drop("income_cat",axis=1)
        print(housing)
```

```
    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)

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


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

    # Construct the full pipeline
    pipeline = build_pipeline(num_attribs,cat_attribs)
    housing_prepared = pipeline.fit_transform(housing_features)
    print(housing_prepared)


    # Train model
    model = RandomForestRegressor(random_state=42)
    model.fit(housing_prepared,housing_labels)

    #Joblib
    joblib.dump(model, MODEL_FILE)
    joblib.dump(pipeline, PIPELINE_FILE)
    print("Model is trained. Congrats!")


    #Interference (IN comming data)
else:
     model = joblib.load(MODEL_FILE)
     pipeline = joblib.load(PIPELINE_FILE)

     input_data = pd.read_csv("input.csv")
     transformed_input = pipeline.transform(input_data)
     predictions = model.predict(transformed_input)
     input_data['median_house_value']=predictions

     input_data.to_csv("output.csv",index=False)
     print("Inference is complete,resltes saved to output.csv")
```

```
Inference is complete,resltes saved to output.csv
```

In [ ]:

In [ ]:

In [ ]: