HousingPriceAnalysis

March 26, 2025

[4]:
```python
import pandas as pd
housing = pd.read_csv("datasets/housing/housing.csv")
```

[6]:
```python
housing.head()
```

[6]:
```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0   
1    -122.22     37.86                21.0       7099.0          1106.0   
2    -122.24     37.85                52.0       1467.0           190.0   
3    -122.25     37.85                52.0       1274.0           235.0   
4    -122.25     37.85                52.0       1627.0           280.0   

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
```

[7]:
```python
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
```

[8]:
```python
housing["ocean_proximity"].value_counts()
```

[8]:
```
ocean_proximity
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: count, dtype: int64
```

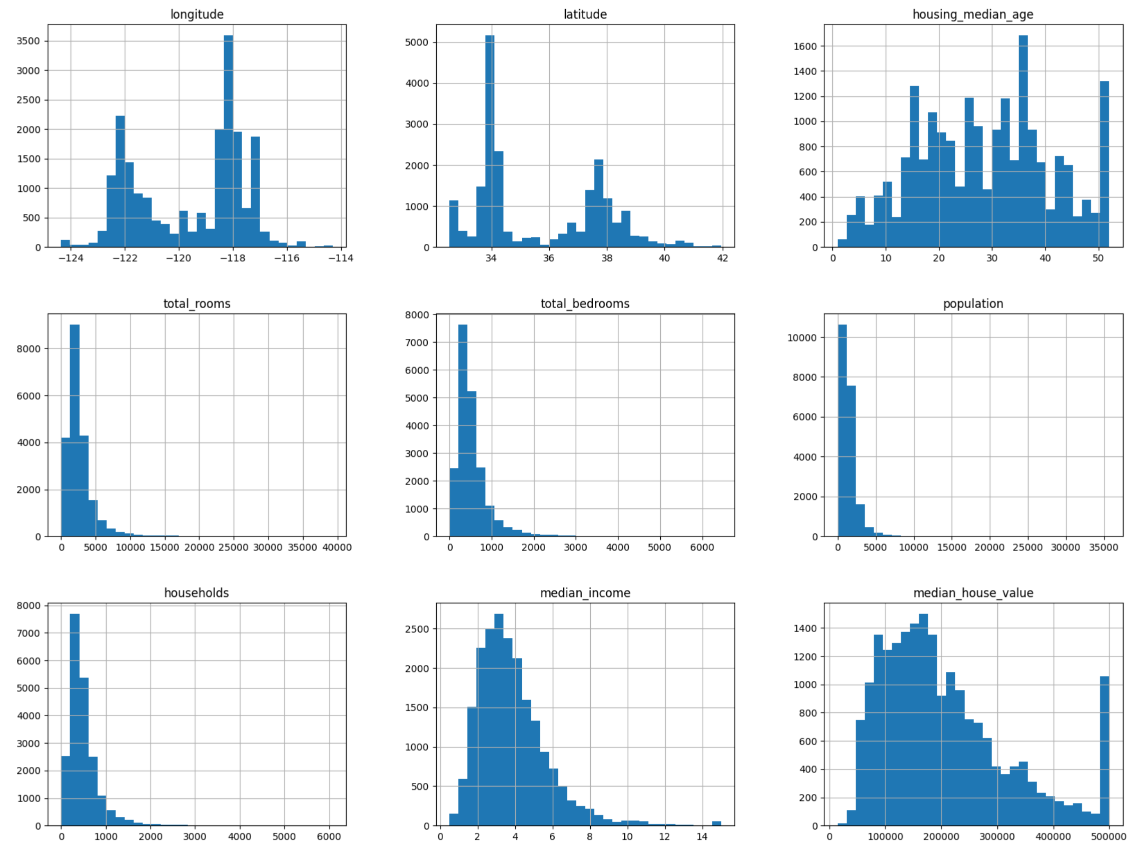
[9]:
```python
housing.describe()
```

[9]:
```
          longitude      latitude  housing_median_age   total_rooms  \
count  20640.000000  20640.000000        20640.000000  20640.000000
mean    -119.569704     35.631861           28.639486   2635.763081
std        2.003532      2.135952           12.585558   2181.615252
min     -124.350000     32.540000            1.000000      2.000000
25%     -121.800000     33.930000           18.000000   1447.750000
50%     -118.490000     34.260000           29.000000   2127.000000
75%     -118.010000     37.710000           37.000000   3148.000000
max     -114.310000     41.950000           52.000000  39320.000000

       total_bedrooms    population    households  median_income  \
count    20433.000000  20640.000000  20640.000000   20640.000000
mean       537.870553   1425.476744    499.539680       3.870671
std        421.385070   1132.462122    382.329753       1.899822
min          1.000000      3.000000      1.000000       0.499900
25%        296.000000    787.000000    280.000000       2.563400
50%        435.000000   1166.000000    409.000000       3.534800
75%        647.000000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
```

[10]:
```python
import matplotlib.pyplot as plt
housing.hist(figsize=(20,15), bins=30)
plt.show()
```

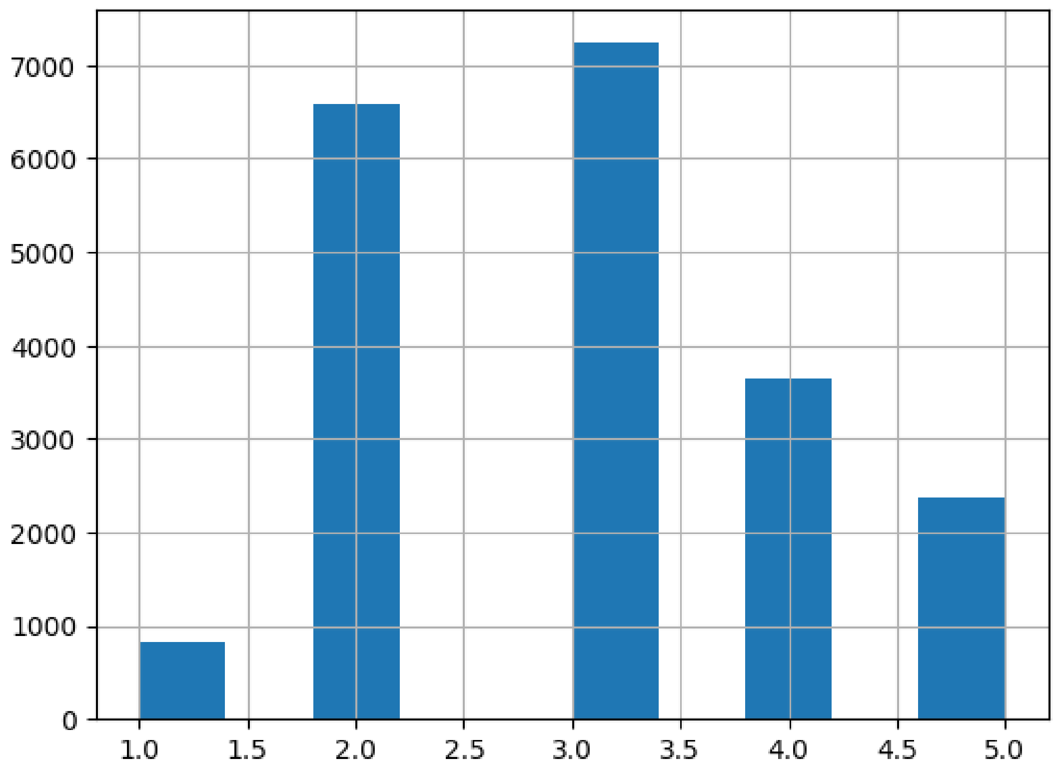


[11]:
```
# import numpy as np
# from sklearn.model_selection import train_test_split

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

# print(len(train_set))
# print(len(test_set))
```

[12]:
```
import numpy as np
housing["income_cat"] = pd.cut(housing["median_income"],
                               bins=[0., 1.5, 3.0, 4.5, 6., np.inf],
                               labels=[1, 2, 3, 4, 5])
housing["income_cat"].hist()
plt.show()
```



[14]:
```
# Creating a test set
from sklearn.model_selection import StratifiedShuffleSplit

stratifiedShuffleSplit = StratifiedShuffleSplit(n_splits=1, test_size=0.2,␣
 ↪random_state=42)

for train_index, test_index in stratifiedShuffleSplit.split(housing,␣
 ↪housing["income_cat"]):
    start_train_set = housing.loc[train_index]
    start_test_index = housing.loc[test_index]

print(start_train_set["income_cat"].value_counts() / len(start_train_set))
print(sum(start_train_set["income_cat"].value_counts()))

print(start_test_index["income_cat"].value_counts() / len(start_test_index))
print(sum(start_test_index["income_cat"].value_counts()))
```

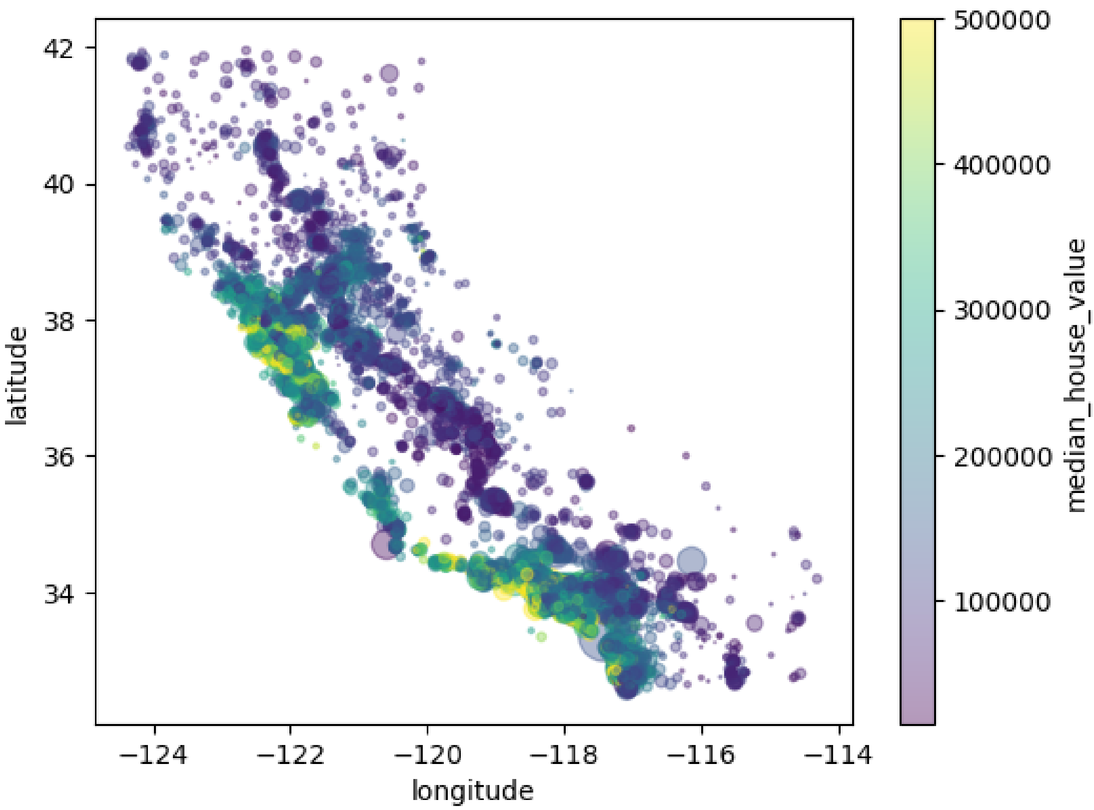
```
income_cat
3    0.350594
2    0.318859
4    0.176296
5    0.114462
```

```
1    0.039789
Name: count, dtype: float64
16512
income_cat
3    0.350533
2    0.318798
4    0.176357
5    0.114341
1    0.039971
Name: count, dtype: float64
4128
```

[15]:
```python
for i in (start_train_set, start_test_index):
    i.drop("income_cat", axis=1, inplace=True)
```

[16]:
```python
# Visualization
housing = start_train_set.copy()
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,␣
 ↪s=housing["population"]/100, c="median_house_value")
```

[16]:
```
<Axes: xlabel='longitude', ylabel='latitude'>
```

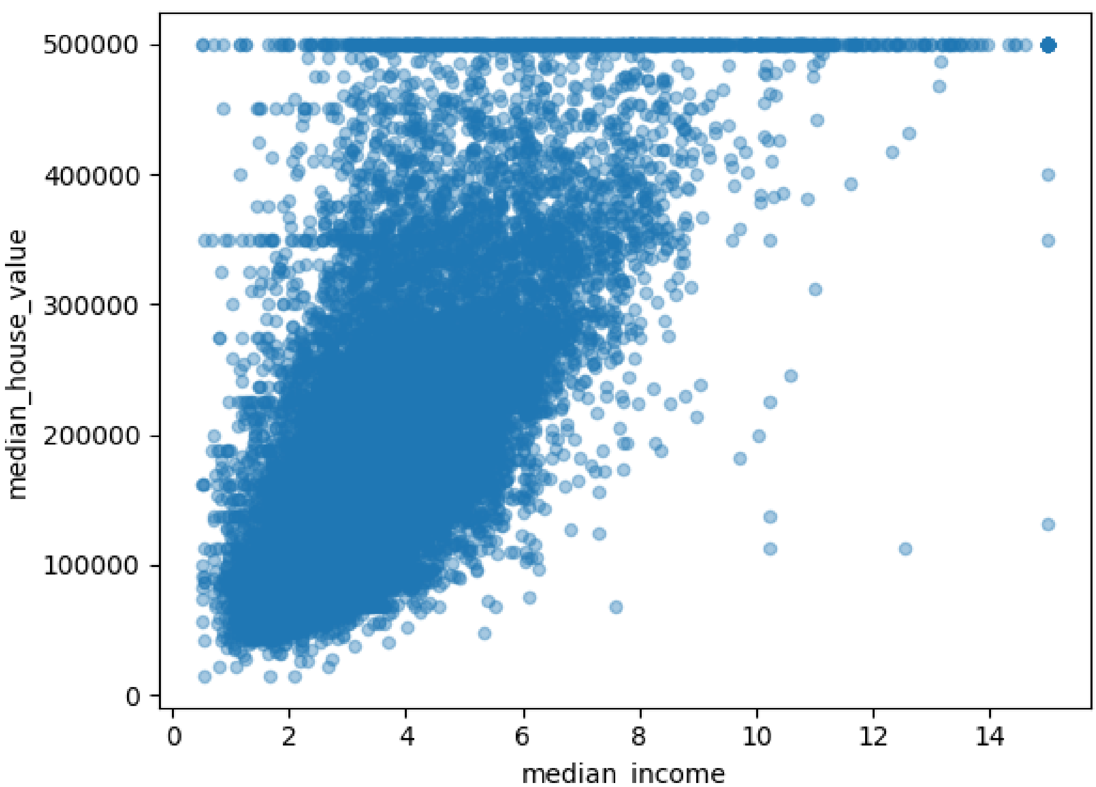


[17]:
```
# Finding relationships
corr_matrix = housing.select_dtypes(include=[np.number]).corr()
corr_matrix["median_house_value"].sort_values()
```

[17]:
```
latitude             -0.142673
longitude            -0.047466
population           -0.026882
total_bedrooms        0.047781
households            0.064590
housing_median_age    0.114146
total_rooms           0.135140
median_income         0.687151
median_house_value    1.000000
Name: median_house_value, dtype: float64
```

[18]:
```
housing.plot(kind="scatter", x="median_income", y="median_house_value", alpha=0.
↪4)
```

[18]:
```
<Axes: xlabel='median_income', ylabel='median_house_value'>
```



[19]:
```python
# Experimenting with attribute combinations


# import itertools

# def create_ratio_columns(data):
#     def create_ratio_column(data, key1, key2):
#         new_column_name = f"{key1}_per_{key2}"
#         data[new_column_name] = data[key1] / data[key2]
#         return new_column_name

#     numeric_columns = data.select_dtypes(include='number').columns.tolist()

#     key_pairs = list(itertools.combinations(numeric_columns, 2))

#     new_column_names = []

#     for key1, key2 in key_pairs:
#         new_column_name = create_ratio_column(data, key1, key2)
#         new_column_names.append(new_column_name)

#     return new_column_names

# new_column_names = create_ratio_columns(data)

# columns_to_drop = data.filter(like='per_median_house_value').columns
# data.drop(columns=columns_to_drop, inplace=True)

housing["rooms_per_household"] = housing["total_rooms"] / housing["households"]
housing["bedrooms_per_room"] = housing["total_bedrooms"] /␣
 ↪housing["total_rooms"]
housing["population_per_household"] = housing["population"] /␣
 ↪housing["households"]

corr_matrix = housing.select_dtypes(include=[np.number]).corr()
corr_matrix["median_house_value"].sort_values()
```

[19]:
```
bedrooms_per_room          -0.259952
latitude                   -0.142673
longitude                  -0.047466
population                 -0.026882
population_per_household   -0.021991
total_bedrooms              0.047781
households                  0.064590
housing_median_age          0.114146
total_rooms                 0.135140
rooms_per_household         0.146255
```

```
median_income                 0.687151
median_house_value            1.000000
Name: median_house_value, dtype: float64
```

[20]:
```
# Data preparation for machine learning algorithms

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

[21]:
```
# Processing text and categorical attributes

from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
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):
        self.add_bedrooms_per_room = add_bedrooms_per_room

    def fit(self, X, y=None):
        return self

    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_household, population_per_household]
        return X

# attr_adder = CombinedAttributesAdder(add_bedrooms_per_room=False)
# housing_extra_attribs = attr_adder.transform(housing.values)
```

[22]:
```
# data = housing.copy()
# median = data["total_bedrooms"].median()
# data["total_bedrooms"] = data["total_bedrooms"].fillna(median)
```

[23]:
```
# from sklearn.impute import SimpleImputer

# imputer = SimpleImputer(strategy='median')
```

```
# data_num = data.drop("ocean_proximity", axis=1)
# imputer.fit(data_num)

# print(imputer.statistics_)
# print(data_num.median().values)
```

[24]:
```
# X = imputer.transform(data_num)
# data_tr = pd.DataFrame(X, columns=data_num.columns, index=data_num.index)
```

[25]:
```
# data_cat = data[["ocean_proximity"]]
# data_cat.head(10)
```

[26]:
```
# from sklearn.preprocessing import OrdinalEncoder

# ordinal_encoder = OrdinalEncoder()
# data_cat_encoded = ordinal_encoder.fit_transform(data_cat)
# print(data_cat_encoded[:10])
# ordinal_encoder.categories_
```

[27]:
```
# from sklearn.preprocessing import OneHotEncoder

# cat_encoder = OneHotEncoder()
# data_cat_1Hot = cat_encoder.fit_transform(data_cat)
# print(data_cat_1Hot.toarray())
```

[28]:
```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")
housing_num = housing.drop("ocean_proximity", axis=1)
imputer.fit(housing_num)
housing_num.median().values
```

[28]:
```
array([-118.51    ,   34.26    ,   29.      , 2119.      ,  433.      ,
       1164.      ,  408.      ,    3.54155])
```

[29]:
```
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)
```

[30]:
```
housing_num_tr
```

[30]:
```
array([[-0.94135046,  1.34743822,  0.02756357, …,  0.01739526,
         0.00622264, -0.12112176],
       [ 1.17178212, -1.19243966, -1.72201763, …,  0.56925554,
        -0.04081077, -0.81086696],
       [ 0.26758118, -0.1259716 ,  1.22045984, …, -0.01802432,
        -0.07537122, -0.33827252],
       …,
       [-1.5707942 ,  1.31001828,  1.53856552, …, -0.5092404 ,
        -0.03743619,  0.32286937],
       [-1.56080303,  1.2492109 , -1.1653327 , …,  0.32814891,
        -0.05915604, -0.45702273],
       [-1.28105026,  2.02567448, -0.13148926, …,  0.01407228,
         0.00657083, -0.12169672]], shape=(16512, 11))
```

[31]:
```python
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

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)
```

[32]:
```python
from sklearn.linear_model import LinearRegression

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

[32]:
```
LinearRegression()
```

[33]:
```python
some_data = housing.iloc[:5]
some_labels = housing_labels.iloc[:5]
some_data_prepared = full_pipeline.transform(some_data)

print("Predictions:", lin_reg.predict(some_data_prepared))
```

```
Predictions: [ 85657.90192014 305492.60737488 152056.46122456 186095.70946094
 244550.67966089]
```

[34]:
```python
print("Labels:", list(some_labels))
```

```
Labels: [72100.0, 279600.0, 82700.0, 112500.0, 238300.0]
```

[35]:
```python
from sklearn.metrics import mean_squared_error

housing_predictions = lin_reg.predict(housing_prepared)
lin_mse = mean_squared_error(housing_labels, housing_predictions)
lin_rmse = np.sqrt(lin_mse)
lin_rmse
```

[35]:
```
np.float64(68627.87390018745)
```

[36]:
```python
from sklearn.tree import DecisionTreeRegressor

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

[36]:
```
DecisionTreeRegressor(random_state=42)
```

[39]:
```python
housing_predictions = tree_reg.predict(housing_prepared)
tree_mse = mean_squared_error(housing_labels, housing_predictions)
tree_rmse = np.sqrt(tree_mse)
```

[40]:
```python
print("Root Mean Squared Error:", tree_rmse)
```

```
Root Mean Squared Error: 0.0
```

[ ]:
```python

```