SevvalTasyonan

March 24, 2023

[16]:
```
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
```

[3]:
```
data = pd.read_csv("housing.csv")
```

[7]:
```
data.columns
```

[7]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity'],
      dtype='object')
```

1-) The task is a regression task where we are supposed to predict values of each house. Our features are 'longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'median_house_value' and 'ocean_proximity'. Our target column is median_house_value. We are going to predict house values using the some of the features after investigate which features are suitable for our regression model.

[5]:
```
data
```

[5]:
```
       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
…            …        …                   …            …              …
20635    -121.09     39.48                25.0       1665.0           374.0
20636    -121.21     39.49                18.0        697.0           150.0
20637    -121.22     39.43                17.0       2254.0           485.0
20638    -121.32     39.43                18.0       1860.0           409.0
20639    -121.24     39.37                16.0       2785.0           616.0

       population  households  median_income  median_house_value  \
0           322.0       126.0         8.3252            452600.0
1          2401.0      1138.0         8.3014            358500.0
```

```
2          496.0       177.0         7.2574            352100.0
3          558.0       219.0         5.6431            341300.0
4          565.0       259.0         3.8462            342200.0
…            …         …            …                 …
20635      845.0       330.0         1.5603             78100.0
20636      356.0       114.0         2.5568             77100.0
20637     1007.0       433.0         1.7000             92300.0
20638      741.0       349.0         1.8672             84700.0
20639     1387.0       530.0         2.3886             89400.0

      ocean_proximity
0            NEAR BAY
1            NEAR BAY
2            NEAR BAY
3            NEAR BAY
4            NEAR BAY
…                 …
20635          INLAND
20636          INLAND
20637          INLAND
20638          INLAND
20639          INLAND

[20640 rows x 10 columns]
```

[6]:
```
data.describe()
```

[6]:
```
          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
```

[8]:
```python
data.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
```

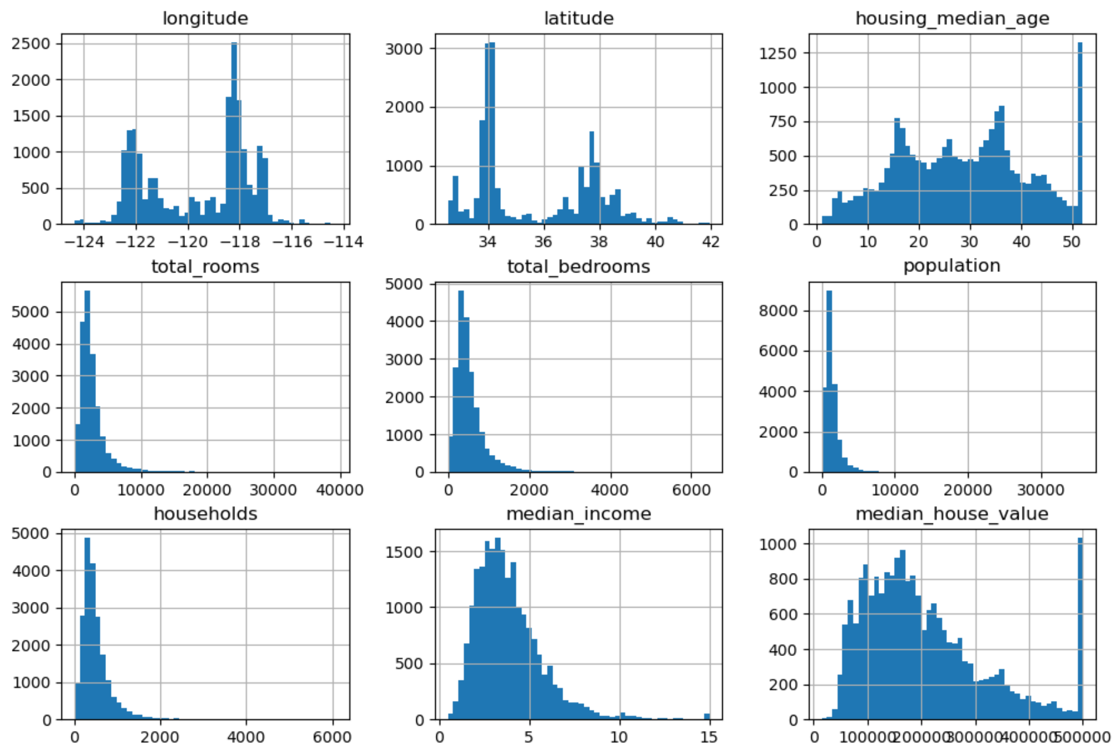
[10]:
```python
data.shape
```

[10]:
```
(20640, 10)
```

[11]:
```python
data.ocean_proximity.value_counts()
```

[11]:
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

[13]:
```python
data.hist(bins=50, figsize=(12, 8))
plt.show()
```



[14]:
```
data.median_income.describe()
```

[14]:
```
count    20640.000000
mean         3.870671
std          1.899822
min          0.499900
25%          2.563400
50%          3.534800
75%          4.743250
max         15.000100
Name: median_income, dtype: float64
```

[18]:
```
data.corr()
```

[18]:
```
                    longitude  latitude  housing_median_age  total_rooms  \
longitude            1.000000 -0.924664           -0.108197     0.044568
latitude            -0.924664  1.000000            0.011173    -0.036100
housing_median_age  -0.108197  0.011173            1.000000    -0.361262
total_rooms          0.044568 -0.036100           -0.361262     1.000000
total_bedrooms       0.069608 -0.066983           -0.320451     0.930380
population           0.099773 -0.108785           -0.296244     0.857126
households           0.055310 -0.071035           -0.302916     0.918484
median_income       -0.015176 -0.079809           -0.119034     0.198050
```

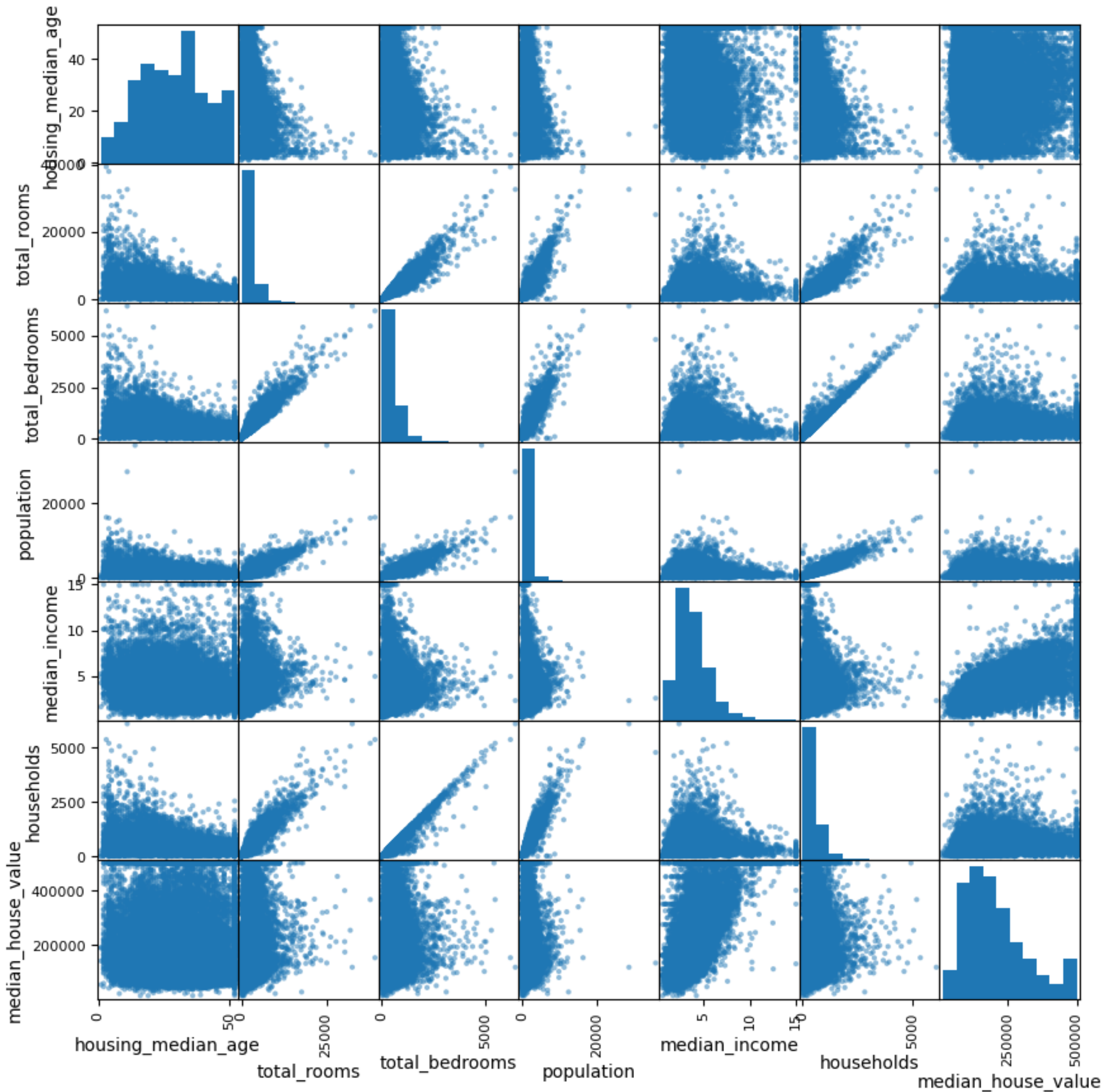
```
median_house_value  -0.045967 -0.144160            0.105623     0.134153

                    total_bedrooms  population  households  median_income  \
longitude                 0.069608    0.099773    0.055310      -0.015176
latitude                 -0.066983   -0.108785   -0.071035      -0.079809
housing_median_age       -0.320451   -0.296244   -0.302916      -0.119034
total_rooms               0.930380    0.857126    0.918484       0.198050
total_bedrooms            1.000000    0.877747    0.979728      -0.007723
population                0.877747    1.000000    0.907222       0.004834
households                0.979728    0.907222    1.000000       0.013033
median_income            -0.007723    0.004834    0.013033       1.000000
median_house_value        0.049686   -0.024650    0.065843       0.688075

                    median_house_value
longitude                    -0.045967
latitude                     -0.144160
housing_median_age            0.105623
total_rooms                   0.134153
total_bedrooms                0.049686
population                   -0.024650
households                    0.065843
median_income                 0.688075
median_house_value            1.000000
```

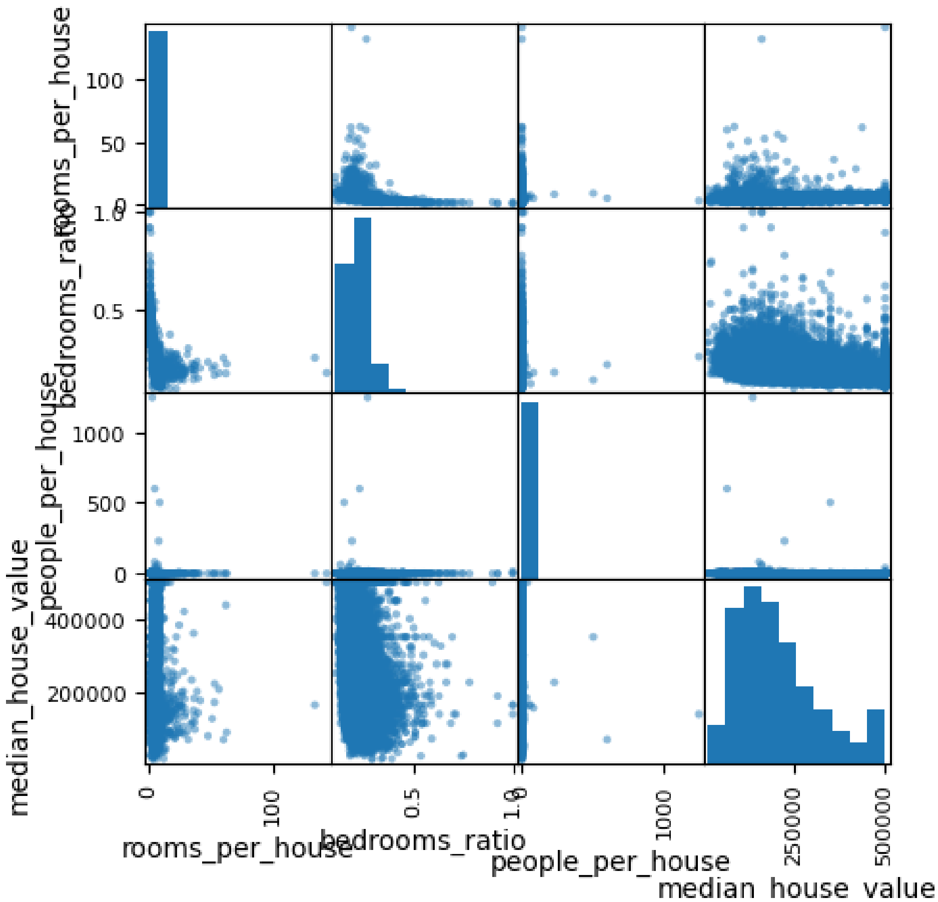
[26]:

```
from pandas.plotting import scatter_matrix
features = ["housing_median_age", "total_rooms", "total_bedrooms",␣
 ↪"population", "median_income", "households","median_house_value"]
scatter_matrix(data[features], figsize = (10,10))
plt.show()
```



[4]:
```
data.total_bedrooms.fillna(value = data.total_bedrooms.mean(), inplace = True)
data["rooms_per_house"] = data["total_rooms"] / data["households"]
data["bedrooms_ratio"] = data["total_bedrooms"] / data["total_rooms"]
data["people_per_house"] = data["population"] / data["households"]
```

[37]:
```
features = ["rooms_per_house", "bedrooms_ratio", "people_per_house",␣
 ↪"median_house_value"]
scatter_matrix(data[features], figsize = (5,5))
plt.show()
```




```
[ ]:
```

2-) Our data consist of 20640 row and 10 feature. Type of the 'ocean_proximity' feature is object while types of the other feautres are numeric(float64). 'total_bedrooms' feature has 167 NA values. 'ocean_proximity' feature has a 5 different value which are '1H OCEAN', 'INLAND', 'NEAR OCEAN', NEAR BY' AND 'ISLAND'.

When we look at the correlation matrix and scatter matrix, it seems that median_house_value and median_income are higly correlated. I created new features which are called rooms_per_house, bedrooms_ratio and people_per_house. It seems that these new features are much more correlated with the target which can be helpful for us.

I also turn ocean_proximity feature into a dummy variable which made 5 new additional dummy variables at the end.

I dropped the "total_rooms", "total_bedrooms", "population" and "ocean_proximity" features because I created new features from them.

```
[5]: data2 = pd.concat([data, pd.get_dummies(data.ocean_proximity)], axis = 1)
data2
```

[5]:
```
       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
…            …         …                   …           …              …
20635    -121.09     39.48                25.0       1665.0           374.0
20636    -121.21     39.49                18.0        697.0           150.0
20637    -121.22     39.43                17.0       2254.0           485.0
20638    -121.32     39.43                18.0       1860.0           409.0
20639    -121.24     39.37                16.0       2785.0           616.0

       population  households  median_income  median_house_value  \
0           322.0       126.0         8.3252            452600.0
1          2401.0      1138.0         8.3014            358500.0
2           496.0       177.0         7.2574            352100.0
3           558.0       219.0         5.6431            341300.0
4           565.0       259.0         3.8462            342200.0
…             …           …             …                  …
20635       845.0       330.0         1.5603             78100.0
20636       356.0       114.0         2.5568             77100.0
20637      1007.0       433.0         1.7000             92300.0
20638       741.0       349.0         1.8672             84700.0
20639      1387.0       530.0         2.3886             89400.0

      ocean_proximity  rooms_per_house  bedrooms_ratio  people_per_house  \
0            NEAR BAY         6.984127        0.146591          2.555556
1            NEAR BAY         6.238137        0.155797          2.109842
2            NEAR BAY         8.288136        0.129516          2.802260
3            NEAR BAY         5.817352        0.184458          2.547945
4            NEAR BAY         6.281853        0.172096          2.181467
…                 …              …               …                …
20635          INLAND         5.045455        0.224625          2.560606
20636          INLAND         6.114035        0.215208          3.122807
20637          INLAND         5.205543        0.215173          2.325635
20638          INLAND         5.329513        0.219892          2.123209
20639          INLAND         5.254717        0.221185          2.616981

       <1H OCEAN  INLAND  ISLAND  NEAR BAY  NEAR OCEAN
0              0       0       0         1           0
1              0       0       0         1           0
2              0       0       0         1           0
3              0       0       0         1           0
4              0       0       0         1           0
…          …     …     …       …         …
20635          0       1       0         0           0
```

```
20636          0       1       0         0           0
20637          0       1       0         0           0
20638          0       1       0         0           0
20639          0       1       0         0           0

[20640 rows x 18 columns]
```

[6]:
```python
data2.drop(columns=["total_rooms", "total_bedrooms", "population",␣
↪"ocean_proximity"], inplace = True)
```

[43]:
```python
data2
```

[43]:
```
       longitude  latitude  housing_median_age  households  median_income  \
0        -122.23     37.88                41.0       126.0         8.3252
1        -122.22     37.86                21.0      1138.0         8.3014
2        -122.24     37.85                52.0       177.0         7.2574
3        -122.25     37.85                52.0       219.0         5.6431
4        -122.25     37.85                52.0       259.0         3.8462
…              …        …                   …           …              …
20635    -121.09     39.48                25.0       330.0         1.5603
20636    -121.21     39.49                18.0       114.0         2.5568
20637    -121.22     39.43                17.0       433.0         1.7000
20638    -121.32     39.43                18.0       349.0         1.8672
20639    -121.24     39.37                16.0       530.0         2.3886

       median_house_value  rooms_per_house  bedrooms_ratio  people_per_house  \
0                452600.0         6.984127        0.146591          2.555556
1                358500.0         6.238137        0.155797          2.109842
2                352100.0         8.288136        0.129516          2.802260
3                341300.0         5.817352        0.184458          2.547945
4                342200.0         6.281853        0.172096          2.181467
…                       …                …               …                 …
20635             78100.0         5.045455        0.224625          2.560606
20636             77100.0         6.114035        0.215208          3.122807
20637             92300.0         5.205543        0.215173          2.325635
20638             84700.0         5.329513        0.219892          2.123209
20639             89400.0         5.254717        0.221185          2.616981

       <1H OCEAN  INLAND  ISLAND  NEAR BAY  NEAR OCEAN
0              0       0       0         1           0
1              0       0       0         1           0
2              0       0       0         1           0
3              0       0       0         1           0
4              0       0       0         1           0
…              …       …       …         …           …
20635          0       1       0         0           0
20636          0       1       0         0           0
```

```
20637          0        1        0          0            0
20638          0        1        0          0            0
20639          0        1        0          0            0

[20640 rows x 14 columns]
```

[ ]:

3-) In the below, I divided the data into train data and test data. Train data has %75 of the data and test data has %25 of the data. I trained a Linear Regreession Model.

[7]:
```
from sklearn.model_selection import train_test_split
train_data, test_data = train_test_split(data2, test_size=0.20, random_state=41)
```

[8]:
```
train_data.shape, test_data.shape
```

[8]:
```
((16512, 14), (4128, 14))
```

[56]:
```
train_data.columns
```

[56]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'households',
       'median_income', 'median_house_value', 'rooms_per_house',
       'bedrooms_ratio', 'people_per_house', '<1H OCEAN', 'INLAND', 'ISLAND',
       'NEAR BAY', 'NEAR OCEAN'],
      dtype='object')
```

[9]:
```
X_train = train_data.loc[:,['longitude', 'latitude', 'housing_median_age',␣
 ↪'households',
                               'median_income', 'rooms_per_house',
                               'bedrooms_ratio', 'people_per_house', '<1H␣
 ↪OCEAN', 'INLAND', 'ISLAND',
                               'NEAR BAY', 'NEAR OCEAN']]
y_train = train_data.loc[:,"median_house_value"]


X_test = test_data.loc[:,['longitude', 'latitude', 'housing_median_age',␣
 ↪'households',
                               'median_income', 'rooms_per_house',
                               'bedrooms_ratio', 'people_per_house', '<1H␣
 ↪OCEAN', 'INLAND', 'ISLAND',
                               'NEAR BAY', 'NEAR OCEAN']]
y_test = test_data.loc[:,"median_house_value"]
```

[10]:
```
import numpy as np
from sklearn.linear_model import LinearRegression
```

[11]:
```
reg = LinearRegression().fit(X_train, y_train)
```

[61]:
```
reg.coef_
```

[61]:
```
array([-2.60503463e+04, -2.40927291e+04,  1.09384440e+03,  2.48037921e+01,
        3.91812402e+04,  2.69777613e+03,  1.46791733e+05, -3.12430361e+02,
       -2.66177252e+04, -6.73270733e+04,  1.36952608e+05, -2.44985665e+04,
       -1.85092432e+04])
```

[ ]:

4-) In the above, I used RMSE as the metric to report the performance of the trained model. Because this is a regression task, RMSE is suitable for this model.

[12]:
```
train_predictions = reg.predict(X_train)
test_predictions = reg.predict(X_test)
```

[13]:
```
from sklearn.metrics import mean_squared_error
train_RMSE = mean_squared_error(y_train, train_predictions)
test_RMSE = mean_squared_error(y_test, test_predictions)
```

[68]:
```
train_RMSE, test_RMSE
```

[68]:
```
(5023037641.406394, 5122944910.218625)
```

5-) Overfitting means, model fits exactly its trained data but shows poor performance on the unseen test data. In overfitting, train error is very small and test error is large.

Underfitting means, models does not fit well to trained data and shows poor performance. Also show poor performance on the test data. In underfitting, train error is large and test error is large also.

When we look at the above RMSEs, train error and test error are close each other. So, i think there is no overfitting or underfitting.

[ ]:

6-) I choose my feature values as follows; longitude = -118, latitude = 39, housing_median_age = 5, households = 1000, median_income = 5, rooms_per_house = 10, bedrooms_ratio = 2.5, people_per_house = 2, <1H OCEAN = 0, INLAND = 0, ISLAND = 0, NEAR BAY = 0, NEAR OCEAN = 1,

[17]:
```
reg.predict( np.array([-118, 39, 5, 1000, 5, 10, 2.5, 2, 0, 0, 0, 0, 1] ).
 ↪reshape(1, -1) )
```

[17]:
```
array([483941.14992896])
```

Prediction of the observation I created is 483941.14992896.