hw

July 17, 2024

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['df_target'] = data.target
print(data.DESCR)
```

```
.. _california_housing_dataset:

California Housing dataset
--------------------------

**Data Set Characteristics:**

:Number of Instances: 20640

:Number of Attributes: 8 numeric, predictive attributes and the target

:Attribute Information:
    - MedInc        median income in block group
    - HouseAge      median house age in block group
    - AveRooms      average number of rooms per household
    - AveBedrms     average number of bedrooms per household
    - Population    block group population
    - AveOccup      average number of household members
    - Latitude      block group latitude
    - Longitude     block group longitude

:Missing Attribute Values: None

This dataset was obtained from the StatLib repository.
https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

The target variable is the median house value for California districts,
```

```
expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per census
block group. A block group is the smallest geographical unit for which the U.S.
Census Bureau publishes sample data (a block group typically has a population
of 600 to 3,000 people).

A household is a group of people residing within a home. Since the average
number of rooms and bedrooms in this dataset are provided per household, these
columns may take surprisingly large values for block groups with few households
and many empty houses, such as vacation resorts.

It can be downloaded/loaded using the
:func:`sklearn.datasets.fetch_california_housing` function.

.. topic:: References

    - Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,
      Statistics and Probability Letters, 33 (1997) 291-297
```

[3]:
```
pd.concat([df.head(10), df.tail(10)])
```

[3]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	\
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	
5	4.0368	52.0	4.761658	1.103627	413.0	2.139896	37.85	
6	3.6591	52.0	4.931907	0.951362	1094.0	2.128405	37.84	
7	3.1200	52.0	4.797527	1.061824	1157.0	1.788253	37.84	
8	2.0804	42.0	4.294118	1.117647	1206.0	2.026891	37.84	
9	3.6912	52.0	4.970588	0.990196	1551.0	2.172269	37.84	
20630	3.5673	11.0	5.932584	1.134831	1257.0	2.824719	39.29	
20631	3.5179	15.0	6.145833	1.141204	1200.0	2.777778	39.33	
20632	3.1250	15.0	6.023377	1.080519	1047.0	2.719481	39.26	
20633	2.5495	27.0	5.445026	1.078534	1082.0	2.832461	39.19	
20634	3.7125	28.0	6.779070	1.148256	1041.0	3.026163	39.27	
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	

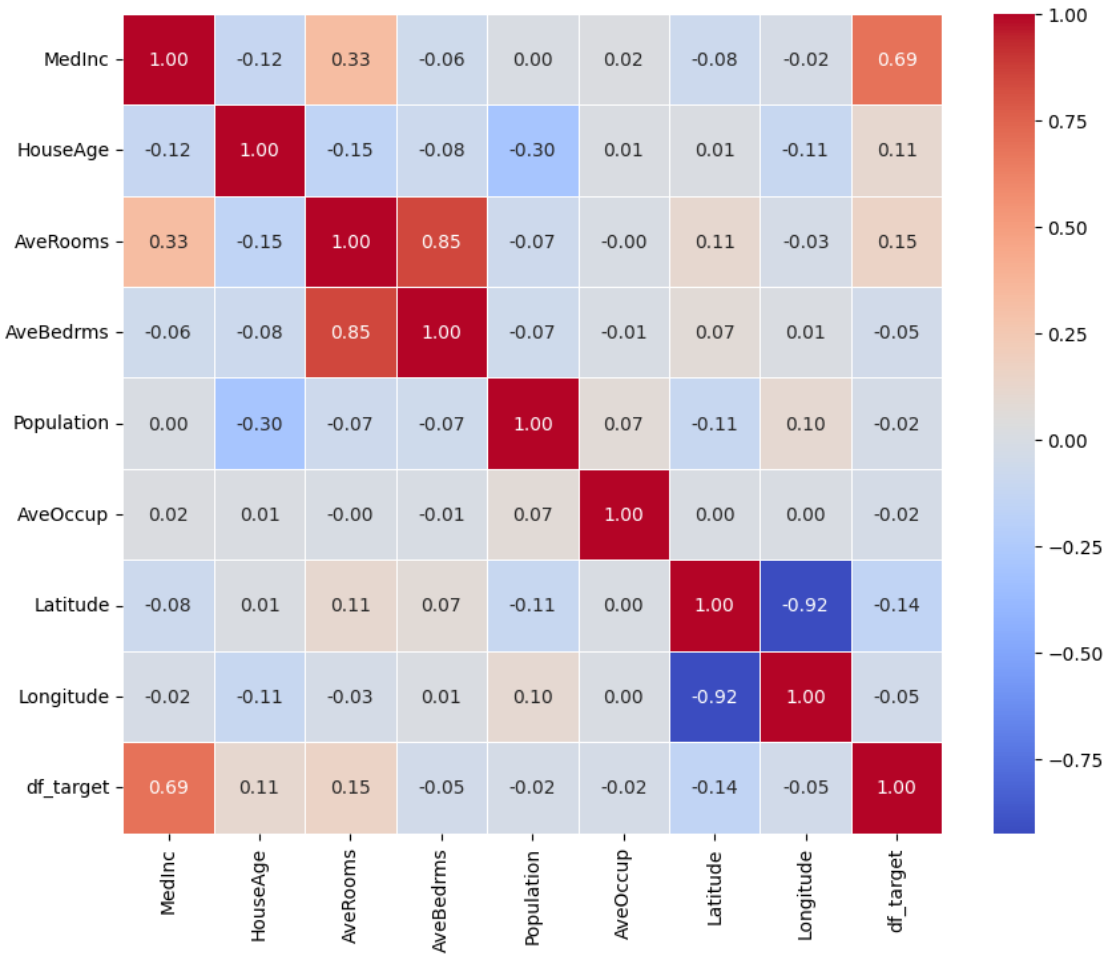
	Longitude	df_target
0	-122.23	4.526

```
1        -122.22      3.585
2        -122.24      3.521
3        -122.25      3.413
4        -122.25      3.422
5        -122.25      2.697
6        -122.25      2.992
7        -122.25      2.414
8        -122.26      2.267
9        -122.25      2.611
20630    -121.32      1.120
20631    -121.40      1.072
20632    -121.45      1.156
20633    -121.53      0.983
20634    -121.56      1.168
20635    -121.09      0.781
20636    -121.21      0.771
20637    -121.22      0.923
20638    -121.32      0.847
20639    -121.24      0.894
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   MedInc      20640 non-null  float64
 1   HouseAge    20640 non-null  float64
 2   AveRooms    20640 non-null  float64
 3   AveBedrms   20640 non-null  float64
 4   Population  20640 non-null  float64
 5   AveOccup    20640 non-null  float64
 6   Latitude    20640 non-null  float64
 7   Longitude   20640 non-null  float64
 8   df_target   20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

[5]:
```
import seaborn as sns

corr = df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, fmt=".2f", cmap='coolwarm', square=True,␣
 ↪linewidths=.5)
plt.show()
```



[6]:
```
from sklearn.model_selection import train_test_split

y = df['df_target']
X = df.drop('df_target', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[7]:
```
(X_train.shape, y_train.shape), (X_test.shape, y_test.shape)
```

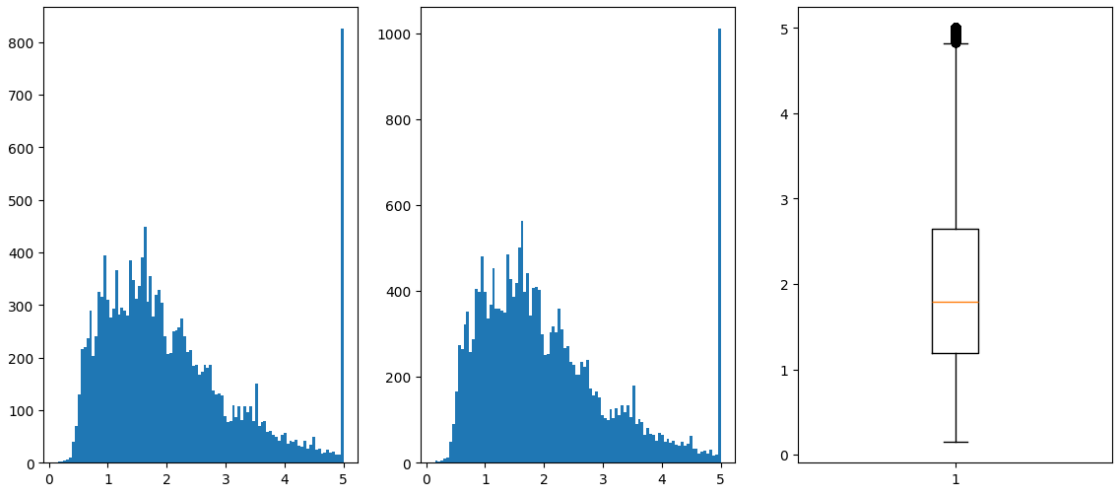
[7]:
```
(((16512, 8), (16512,)), ((4128, 8), (4128,)))
```

[8]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

def get_r2_rmse(X, y, model=None):
    if model is None:
```

```
        model = LinearRegression()
    model.fit(X, y)
    pred = model.predict(X_test)
    return (r2_score(y_test, pred), np.sqrt(mean_squared_error(y_test, pred)))

res = get_r2_rmse(X_train, y_train)
print(f"Root mean squared error: {res[1]}")
print(f"R2 score: {res[0]}")
```

```
Root mean squared error: 0.7455813830127761
R2 score: 0.5757877060324511
```

[9]:
```
plt.figure(figsize=(14, 6))
plt.subplot(1, 3, 1)
plt.hist(y_train, bins=100)
plt.subplot(1, 3, 2)
plt.hist(y, bins=100)
plt.subplot(1, 3, 3)
plt.boxplot(y)
plt.subplots_adjust(wspace=0.2)
plt.show()
```



[10]:
```
y_train.value_counts()
```

[10]:
```
df_target
5.00001    786
1.37500     93
1.62500     89
1.12500     82
1.87500     73
```

```
          …
2.74800      1
4.29000      1
3.10500      1
4.18400      1
4.41400      1
Name: count, Length: 3675, dtype: int64
```

target > 5

[11]:
```
df[df['df_target'] > 5]
```

[11]:
```
        MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
89      1.2434      52.0  2.929412   0.917647       396.0  4.658824     37.80
459     1.1696      52.0  2.436000   0.944000      1349.0  5.396000     37.87
493     7.8521      52.0  7.794393   1.051402       517.0  2.415888     37.86
494     9.3959      52.0  7.512097   0.955645      1366.0  2.754032     37.85
509     7.8772      52.0  8.282548   1.049861       947.0  2.623269     37.83
…          …         …         …          …           …         …         …
20422   5.1457      35.0  6.958333   1.217593       576.0  2.666667     34.14
20426  10.0472      11.0  9.890756   1.159664       415.0  3.487395     34.18
20427   8.6499       4.0  7.236059   1.032528      5495.0  2.553439     34.19
20436  12.5420      10.0  9.873315   1.102426      1179.0  3.177898     34.21
20443   3.3438      50.0  5.342857   0.942857       130.0  3.714286     34.27

       Longitude  df_target
89       -122.27    5.00001
459      -122.25    5.00001
493      -122.24    5.00001
494      -122.24    5.00001
509      -122.23    5.00001
…           …          …
20422    -118.90    5.00001
20426    -118.69    5.00001
20427    -118.80    5.00001
20436    -118.69    5.00001
20443    -118.85    5.00001

[965 rows x 9 columns]
```

[12]:
```
corr = X_train.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, fmt=".2f", cmap='coolwarm', square=True,␣
 ↪linewidths=.5)
plt.show()
```

, .

[13]:
```
df.drop(df[df['df_target'] > 5].index, inplace=True)
```

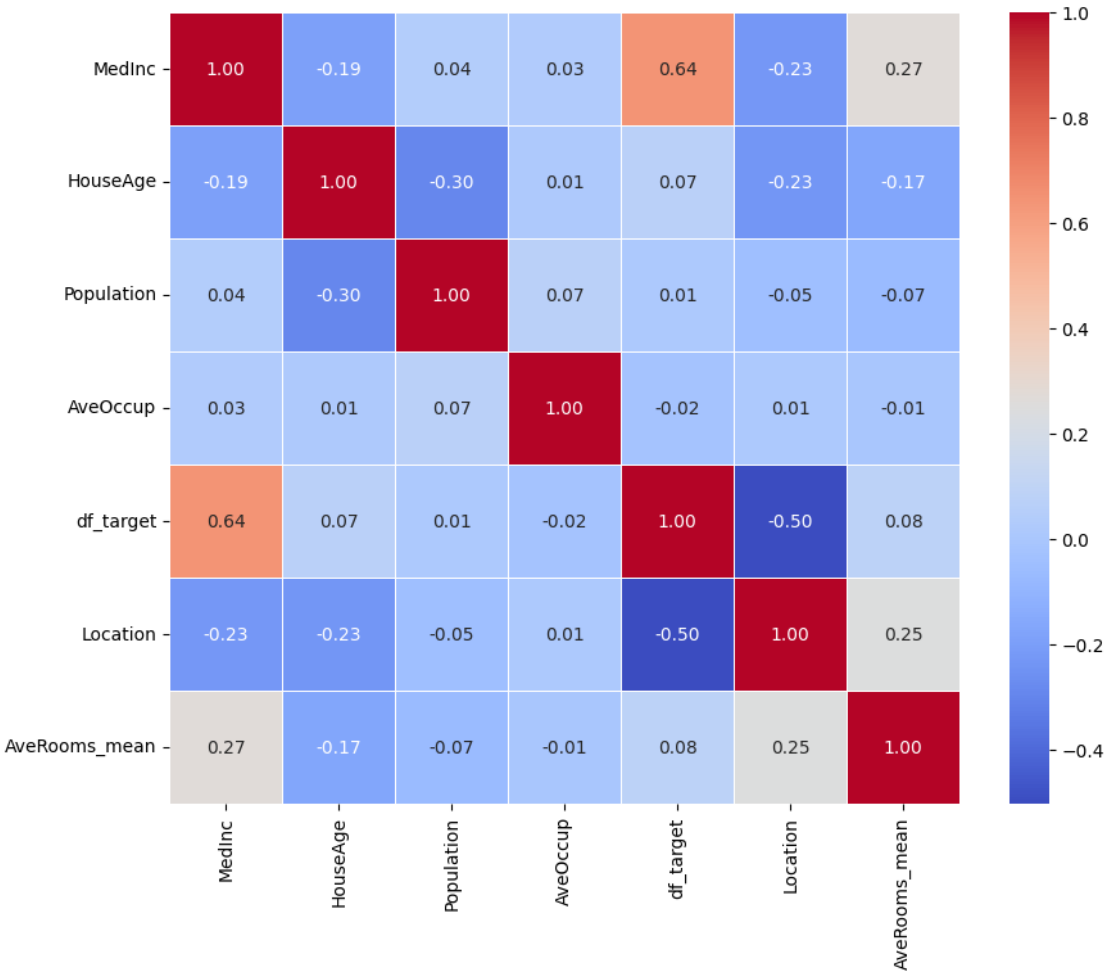
[14]:
```
def attr_to_mean(cols):
    return cols.mean()

df['Location'] = df[['Latitude', 'Longitude']].apply(attr_to_mean, axis=1)
df.drop(['Latitude', 'Longitude'], axis=1, inplace=True)
df['AveRooms_mean'] = df[['AveRooms', 'AveBedrms']].apply(attr_to_mean, axis=1)
df.drop(['AveRooms', 'AveBedrms'], axis=1, inplace=True)
df
```

[14]:

	MedInc	HouseAge	Population	AveOccup	df_target	Location	\
0	8.3252	41.0	322.0	2.555556	4.526	-42.175	
1	8.3014	21.0	2401.0	2.109842	3.585	-42.180	
2	7.2574	52.0	496.0	2.802260	3.521	-42.195	
3	5.6431	52.0	558.0	2.547945	3.413	-42.200	

```
4      3.8462      52.0       565.0  2.181467      3.422   -42.200
…       …          …          …      …             …       …
20635  1.5603      25.0       845.0  2.560606      0.781   -40.805
20636  2.5568      18.0       356.0  3.122807      0.771   -40.860
20637  1.7000      17.0      1007.0  2.325635      0.923   -40.895
20638  1.8672      18.0       741.0  2.123209      0.847   -40.945
20639  2.3886      16.0      1387.0  2.616981      0.894   -40.935

       AveRooms_mean
0           4.003968
1           3.605009
2           4.680791
3           3.445205
4           3.681467
…                …
20635       3.089394
20636       3.714912
20637       3.162818
20638       3.250716
20639       3.208491

[19675 rows x 7 columns]
```

[15]:
```
corr = df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, fmt=".2f", cmap='coolwarm', square=True,␣
 ↪linewidths=.5)
plt.show()
```



[16]:
```
y = df['df_target']
X = df.drop('df_target', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[17]:
```
(X_train.shape, y_train.shape), (X_test.shape, y_test.shape)
```

[17]:
```
(((15740, 6), (15740,)), ((3935, 6), (3935,)))
```

[18]:
```
res = get_r2_rmse(X_train, y_train)
print(f"Root mean squared error: {res[1]}")
print(f"R2 score: {res[0]}")
```

```
Root mean squared error: 0.6635821695296633
R2 score: 0.5583773722378976
```

[19]:
```
df = pd.DataFrame(data.data, columns=data.feature_names)
df['df_target'] = data.target
df
```

[19]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	\
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	
…	…	…	…	…	…	…	…	
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	

	Longitude	df_target
0	-122.23	4.526
1	-122.22	3.585
2	-122.24	3.521
3	-122.25	3.413
4	-122.25	3.422
…	…	…
20635	-121.09	0.781
20636	-121.21	0.771
20637	-121.22	0.923
20638	-121.32	0.847
20639	-121.24	0.894

[20640 rows x 9 columns]

. , , , .

[20]:
```
import functools

def attr_to_pow(cols):
    return cols ** 2

def attr_to_multiply(cols):
    return functools.reduce(lambda x, y: x * y, cols)

def attr_divide(cols):
    return functools.reduce(lambda x, y: x / y, cols)

def attr_to_1(col):
    return 1 / (1 + col)
```

```
df.drop(df[df['df_target'] > 5].index, inplace=True)

df['AveOccup'] = df['AveOccup'].apply(attr_to_pow)
df['MedInc_vs'] = df[['MedInc', 'AveBedrms', 'HouseAge']].
 ↪apply(attr_to_multiply, axis=1)
df['MedInc_vs'] = df[['MedInc_vs', 'Population']].apply(attr_divide, axis=1)
df['Latitude'] = df['Latitude'].apply(attr_to_1)
df['Longitude'] = df['Longitude'].apply(attr_to_1)

pd.concat([df.head(10), df.tail(10)])
```

[20]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	\
0	8.3252	41.0	6.984127	1.023810	322.0	6.530864	0.025720	
1	8.3014	21.0	6.238137	0.971880	2401.0	4.451433	0.025733	
2	7.2574	52.0	8.288136	1.073446	496.0	7.852660	0.025740	
3	5.6431	52.0	5.817352	1.073059	558.0	6.492025	0.025740	
4	3.8462	52.0	6.281853	1.081081	565.0	4.758799	0.025740	
5	4.0368	52.0	4.761658	1.103627	413.0	4.579156	0.025740	
6	3.6591	52.0	4.931907	0.951362	1094.0	4.530106	0.025747	
7	3.1200	52.0	4.797527	1.061824	1157.0	3.197851	0.025747	
8	2.0804	42.0	4.294118	1.117647	1206.0	4.108286	0.025747	
9	3.6912	52.0	4.970588	0.990196	1551.0	4.718752	0.025747	
20630	3.5673	11.0	5.932584	1.134831	1257.0	7.979038	0.024820	
20631	3.5179	15.0	6.145833	1.141204	1200.0	7.716049	0.024795	
20632	3.1250	15.0	6.023377	1.080519	1047.0	7.395574	0.024839	
20633	2.5495	27.0	5.445026	1.078534	1082.0	8.022834	0.024882	
20634	3.7125	28.0	6.779070	1.148256	1041.0	9.157661	0.024832	
20635	1.5603	25.0	5.045455	1.133333	845.0	6.556703	0.024704	
20636	2.5568	18.0	6.114035	1.315789	356.0	9.751924	0.024697	
20637	1.7000	17.0	5.205543	1.120092	1007.0	5.408579	0.024734	
20638	1.8672	18.0	5.329513	1.171920	741.0	4.508017	0.024734	
20639	2.3886	16.0	5.254717	1.162264	1387.0	6.848590	0.024771	

	Longitude	df_target	MedInc_vs
0	-0.008249	4.526	1.085280
1	-0.008249	3.585	0.070565
2	-0.008248	3.521	0.816739
3	-0.008247	3.413	0.564301
4	-0.008247	3.422	0.382688
5	-0.008247	2.697	0.560935
6	-0.008247	2.992	0.165465
7	-0.008247	2.414	0.148894
8	-0.008247	2.267	0.080975
9	-0.008247	2.611	0.122541
20630	-0.008311	1.120	0.035427
20631	-0.008306	1.072	0.050183
20632	-0.008302	1.156	0.048376

```
20633  -0.008297      0.983   0.068616
20634  -0.008295      1.168   0.114660
20635  -0.008327      0.781   0.052318
20636  -0.008319      0.771   0.170101
20637  -0.008318      0.923   0.032146
20638  -0.008311      0.847   0.053155
20639  -0.008317      0.894   0.032025
```

[21]:
```python
for col in df.columns:
    if col == 'df_target':
        continue
    print("Drop col: ", col)
    temp_df = df.drop(col, axis=1)
    y = temp_df['df_target']
    X = temp_df.drop('df_target', axis=1)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
    res = get_r2_rmse(X_train, y_train)
    print(f"Root mean squared error: {res[1]}")
    print(f"R2 score: {res[0]}")
    print()
```

```
Drop col:  MedInc
Root mean squared error: 0.790275272818049
R2 score: 0.37364768299753526

Drop col:  HouseAge
Root mean squared error: 0.6564299267259952
R2 score: 0.567845891390352

Drop col:  AveRooms
Root mean squared error: 0.6588726279242095
R2 score: 0.5646236511969148

Drop col:  AveBedrms
Root mean squared error: 0.6637528950866795
R2 score: 0.5581501028159478

Drop col:  Population
Root mean squared error: 0.6495656278818449
R2 score: 0.576836721326256

Drop col:  AveOccup
Root mean squared error: 0.6496069774109321
R2 score: 0.5767828448325709

Drop col:  Latitude
Root mean squared error: 0.7014389185391859
```

```
R2 score: 0.5065517436787831

Drop col:  Longitude
Root mean squared error: 0.7027381890712674
R2 score: 0.5047220289730183

Drop col:  MedInc_vs
Root mean squared error: 0.6510833045637414
R2 score: 0.574857012807308
```

0.0.1

1.
   - “California Housing” , , ,
     , , , , . -
     .
   - 20640 , .
2.
   - .
     (MedInc) (df_target), .
3.
   - (80%) (20%) `train_test_split`
     `sklearn`.
4.
   - .
   - , (R^2 = 0.5758)
     (RMSE = 0.7456).
5.
   - , ,
     .
6.
   - 5 ( ), .
   - , : (R^2 = 0.5584), (RMSE = 0.64).
7.
   - , “Location” ( ) “AveRooms_mean” (
).
   - .
8.
   - . ,
     .

:)