labset-2

February 28, 2025

Develop a program to compute the correlation matrix to understand the relationships between pairs of features. Visualize the correlation matrix using a heatmap to know which variables have strong positive/negative correlations. Create a pair plot to visualize pairwise relationship between features. Use california housing dataset.

[27]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[28]:

```
df = pd.read_csv('./housing.csv')
print(df)
```

```
       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]
```

[29]:

```
df.shape #.shape is an attribute of a DataFrame that returns the number of rows and columns
```

[29]:

```
(20640, 10)
```

[30]:

```
df.info()  #info() is a method that provides a concise summary of a DataFrame
```

```
<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
```

[31]:

```
df.isnull().sum()
```

[31]:

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
```

```
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

```
[32]: df.duplicated().sum()#duplicated().sum() is used to count the number of␣
        ↪duplicate rows in a DataFrame.
```

```
[32]: 0
```

```
[33]: df.nunique()#nunique() is a method used to count the number of unique values in␣
        ↪each column of a DataFrame or a specific column.
```

```
[33]: longitude               844
      latitude                862
      housing_median_age       52
      total_rooms            5926
      total_bedrooms         1923
      population             3888
      households             1815
      median_income         12928
      median_house_value     3842
      ocean_proximity           5
      dtype: int64
```

```
[34]: df.fillna({'total_bedrooms':0},inplace = True)#fills missing values in the␣
        ↪"total_bedrooms" column with 0 and updates the DataFrame in place.
```

```
[35]: df.isnull().sum()#.isnull().sum() is a powerful method used to identify missing␣
        ↪values  in a DataFrame
```

```
[35]: longitude             0
      latitude              0
      housing_median_age    0
      total_rooms           0
      total_bedrooms        0
      population            0
      households            0
      median_income         0
      median_house_value    0
      ocean_proximity       0
      dtype: int64
```

```
[36]: numerical = df.select_dtypes(include = [np.number]).columns
      print(numerical)
```

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

[37]:

```
correlation_mat = df[numerical].corr()
print(correlation_mat)
```

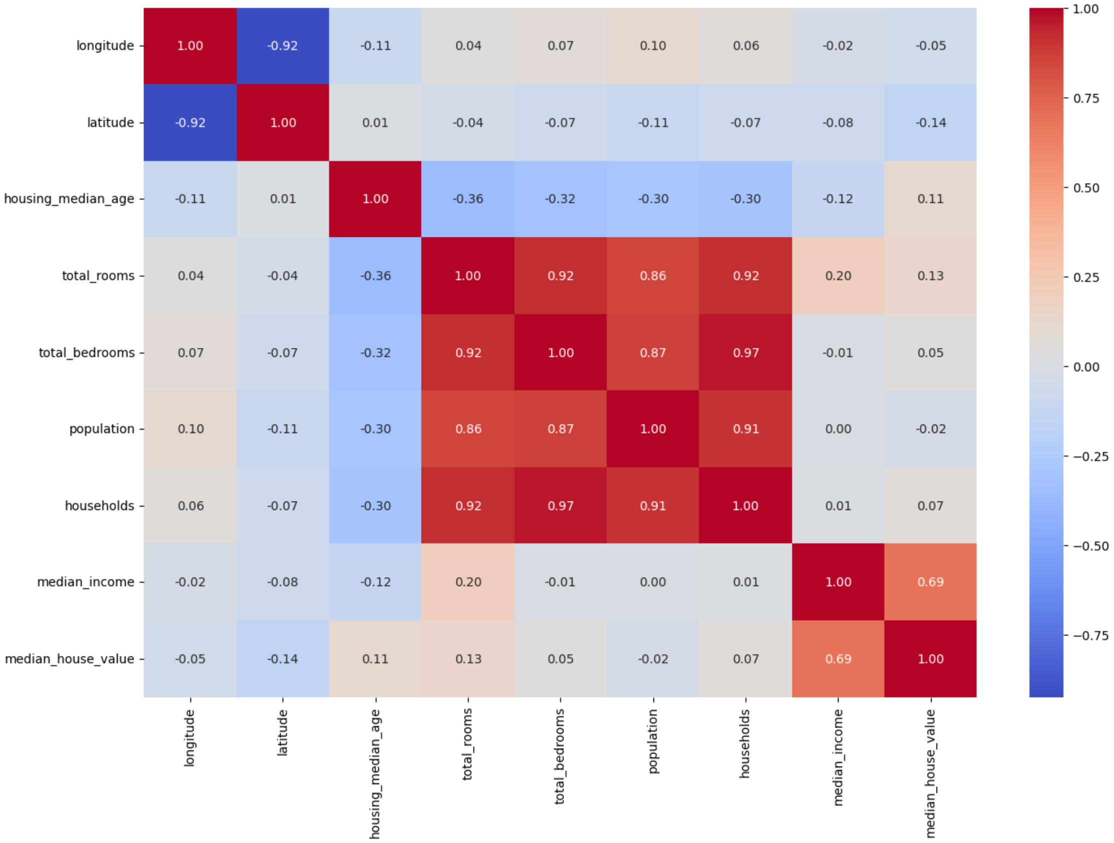
```
                    longitude  latitude  housing_median_age  total_rooms  \
longitude            1.000000 -0.924664           -0.108197     0.044568
latitude            -0.924664  1.000000            0.011173    -0.036100
housing_median_age  -0.108197  0.011173            1.000000    -0.361262
total_rooms          0.044568 -0.036100           -0.361262     1.000000
total_bedrooms       0.068082 -0.065318           -0.317063     0.920196
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                 0.068082    0.099773    0.055310      -0.015176
latitude                 -0.065318   -0.108785   -0.071035      -0.079809
housing_median_age       -0.317063   -0.296244   -0.302916      -0.119034
total_rooms               0.920196    0.857126    0.918484       0.198050
total_bedrooms            1.000000    0.866266    0.966507      -0.007295
population                0.866266    1.000000    0.907222       0.004834
households                0.966507    0.907222    1.000000       0.013033
median_income            -0.007295    0.004834    0.013033       1.000000
median_house_value        0.049148   -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                0.049148
population                   -0.024650
households                    0.065843
median_income                 0.688075
median_house_value            1.000000
```

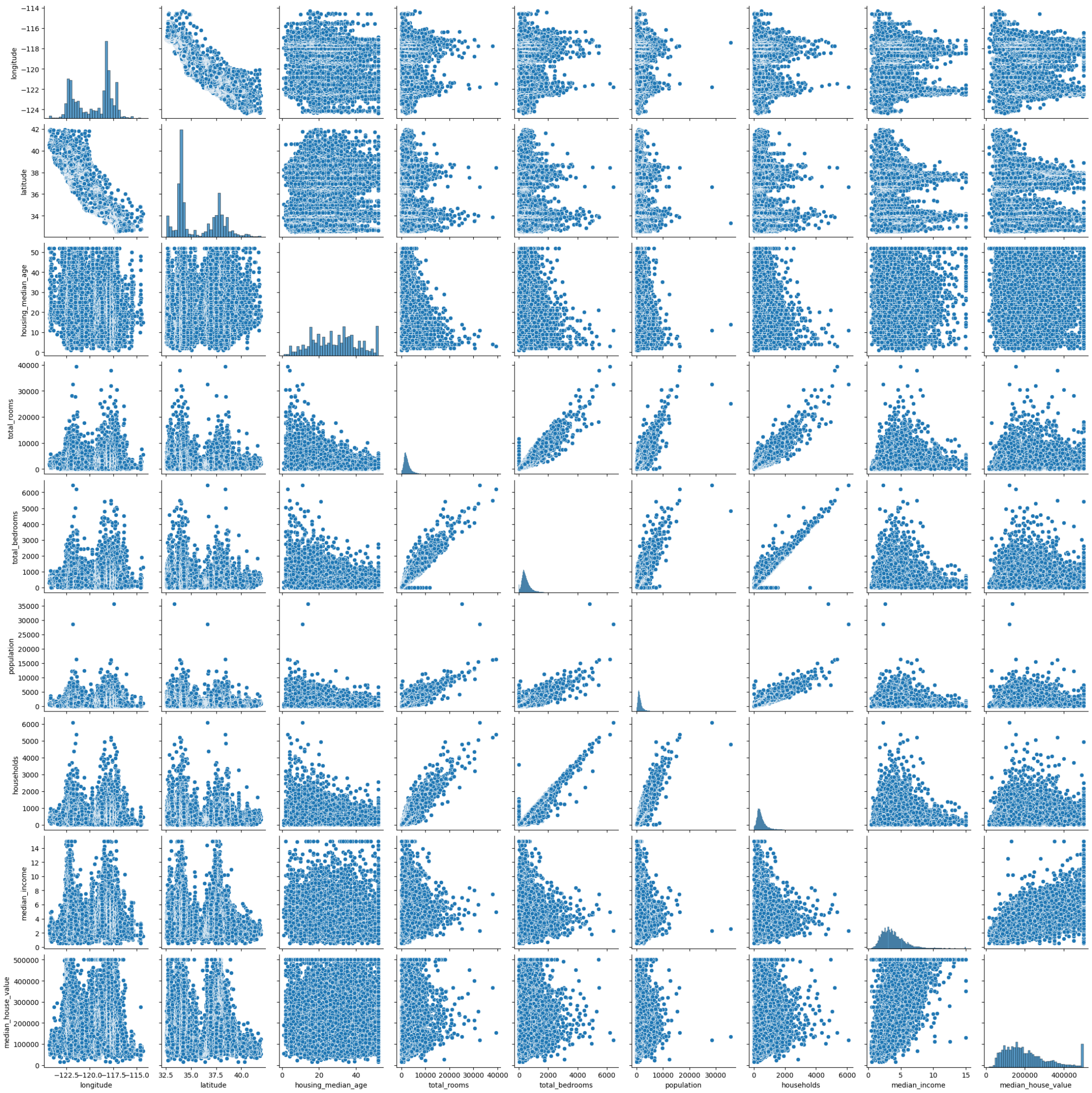
[38]:

```
plt.figure(figsize = (15,10))
sns.heatmap(correlation_mat,cmap = 'coolwarm',fmt = '.2f',annot = True)
plt.show()
```



[40]:

```
sns.pairplot(df[numerical],diag_kind='hist')
plt.show()
```


longitude
latitude
housing_median_age
total_rooms
total_bedrooms
population
households
median_income
median_house_value
−114
−116
−118
−120
−122
−124
42
40
38
36
34
50
40
30
20
10
0
40000
30000
20000
10000
6000
5000
4000
3000
2000
1000
35000
30000
25000
15000
5000
14
12
8
6
4
2
500000
400000
300000
200000
100000
−122.5
−120.0
−117.5
−115.0
32.5
35.0
37.5
40.0
15

[ ]:

[ ]: