شناخت داده و پیش پردازش داده ها

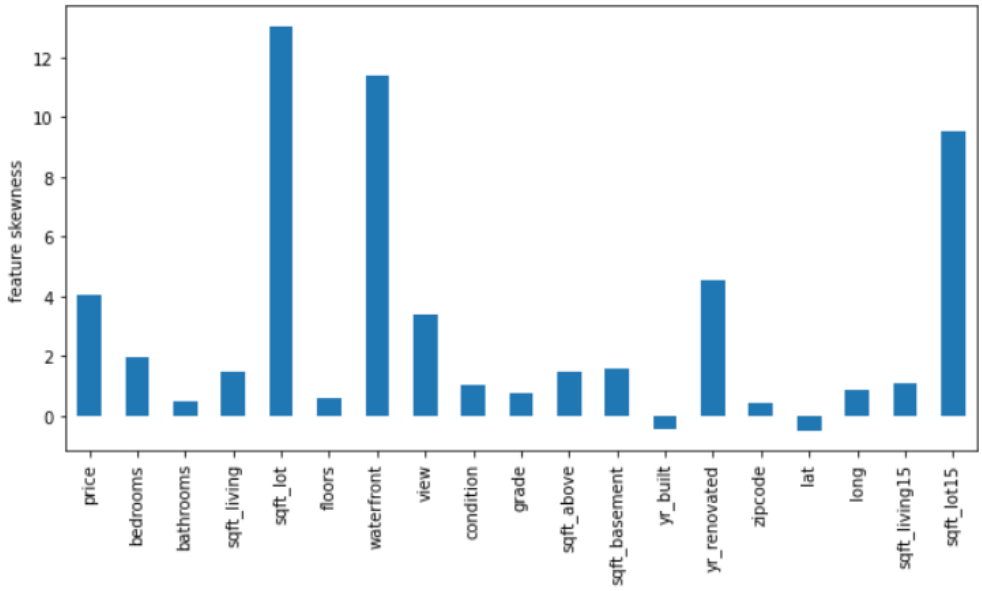
1. تحلیل دیتاست

در ابتدا دیتاست مورد نظر خوانده شده و در یک dataframe برای ادامه تحلیل و بررسی ذخیره میشود.

a. خصوصیات خواسته شده برای هر ویژگی در جدول زیر قابل مشاهده میباشد.

	mean	min	50%	max	mode_1	mode_2	mode_3	mode_4	type
price	540088	75000	450000	7.7e+06	350000	450000			float64
bedrooms	3.37084	0	3	33	3				int64
bathrooms	2.11476	0	2.25	8	2.5				float64
sqft_living	2079.9	290	1910	13540	1300				int64
sqft_lot	15107	520	7618	1.65136e+06	5000				int64
floors	1.49431	1	1.5	3.5	1				float64
waterfront	0.00754176	0	0	1	0				int64
view	0.234303	0	0	4	0				int64
condition	3.40943	1	3	5	3				int64
grade	7.65687	1	7	13	7				int64
sqft_above	1788.39	290	1560	9410	1300				int64
sqft_basement	291.509	0	0	4820	0				int64
yr_built	1971.01	1900	1975	2015	2014				int64
yr_renovated	84.4023	0	0	2015	0				int64
zipcode	98077.9	98001	98065	98199	98103				int64
lat	47.5601	47.1559	47.5718	47.7776	47.5322	47.5491	47.6624	47.6846	float64
long	-122.214	-122.519	-122.23	-121.315	-122.29				float64
sqft_living15	1986.55	399	1840	6210	1540				int64
sqft_lot15	12768.5	651	7620	871200	5000				int64
date					20140623T000000				object

b. کشیدگی برای هر ویژگی حساب شده و در جدول زیر نمایش داده شده اند.

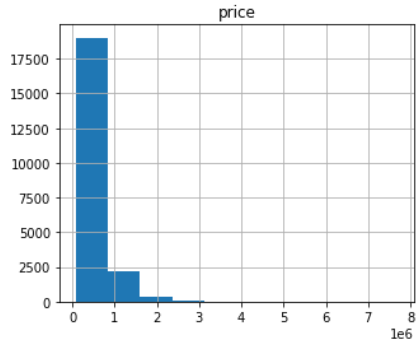


price	4.024069
bedrooms	1.974300
bathrooms	0.511108
sqft_living	1.471555
sqft_lot	13.060019
floors	0.616177
waterfront	11.385108
view	3.395750
condition	1.032805
grade	0.771103
sqft_above	1.446664
sqft_basement	1.577965
yr_built	-0.469805
yr_renovated	4.549493
zipcode	0.405661
lat	-0.485270
long	0.885053
sqft_living15	1.108181
sqft_lot15	9.506743

```
date   => has NO missing value!
price   => has NO missing value!
bedrooms   => has NO missing value!
bathrooms   => has NO missing value!
sqft_living   => has NO missing value!
sqft_lot   => has NO missing value!
floors   => has NO missing value!
waterfront   => has NO missing value!
view   => has NO missing value!
condition   => has NO missing value!
grade   => has NO missing value!
sqft_above   => has NO missing value!
sqft_basement   => has NO missing value!
yr_built   => has NO missing value!
yr_renovated   => has NO missing value!
zipcode   => has NO missing value!
lat   => has NO missing value!
long   => has NO missing value!
sqft_living15   => has NO missing value!
sqft_lot15   => has NO missing value!
```

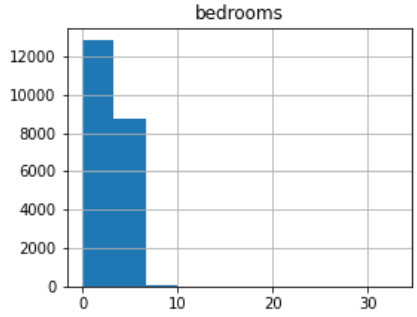
c. در این مرحله کامل بودن داده ها بررسی شده و همانگونه که مشخص میباشد، داده ی نامعتبر یا از دست رفته در هیچ یک از ستون های دیتاست وجود ندارد.

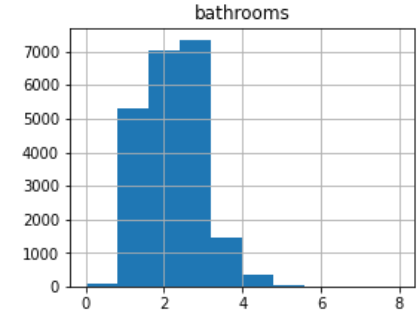
d. در این قسمت نمودارهای هیستوگرام برای تمامی ویژگی ها بجز date به صورت جداگانه رسم شده است.

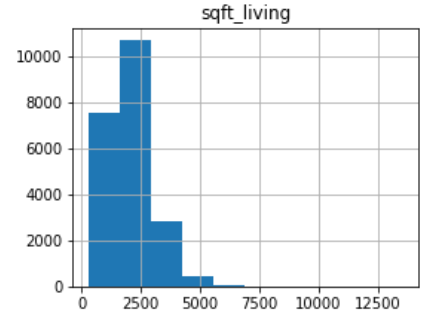


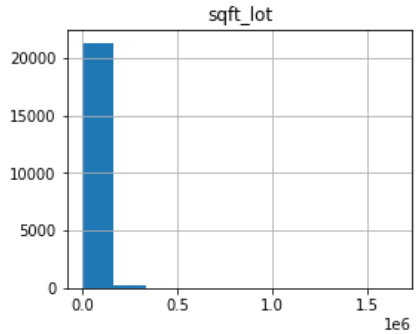
از آنجا که در این مرحله نوع متغیر date از جنس رشته کاراکتر می باشد، هیستوگرام و مفهوم فرکانس برای آن معنی ندارد.

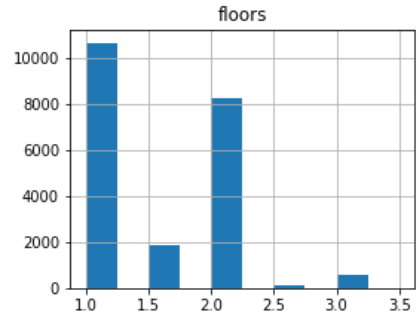
در قسمت a از سوال 2 این مورد بررسی شده و date به نوع datetime تبدیل میشود.

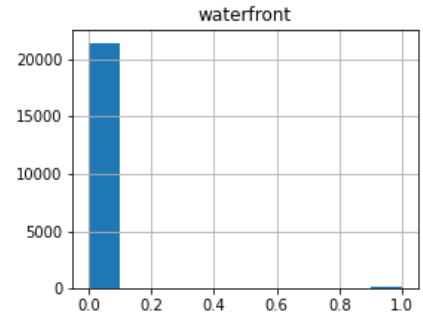

bedrooms
0 2000 4000 6000 8000 10000 12000
0 10 20 30

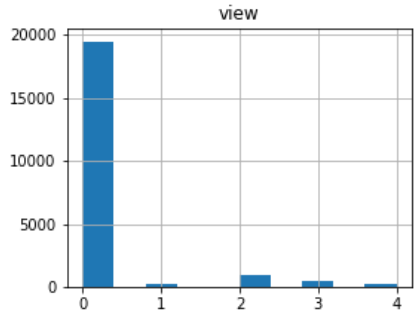

bathrooms
0 1000 2000 3000 4000 5000 6000 7000
0 2 4 6 8

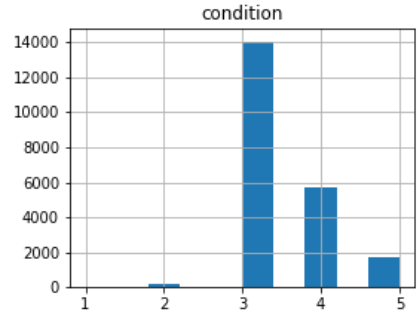

sqft_living
0 2000 4000 6000 8000 10000
0 2500 5000 7500 10000 12500

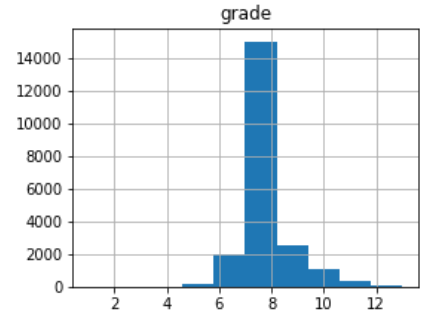

sqft_lot
0 5000 10000 15000 20000
0.0 0.5 1.0 1.5
1e6

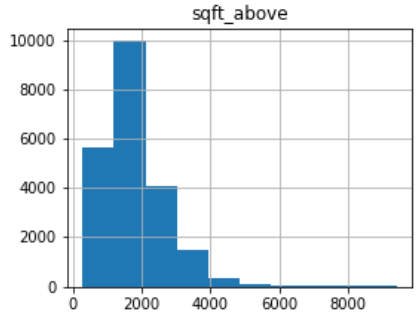

floors
0 2000 4000 6000 8000 10000
1.0 1.5 2.0 2.5 3.0 3.5

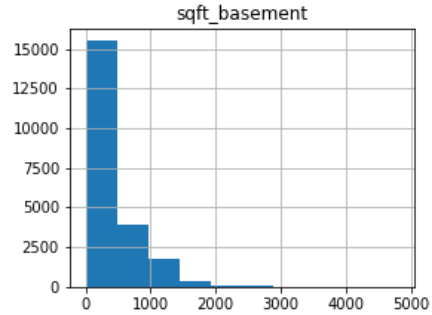

waterfront
0 5000 10000 15000 20000
0.0 0.2 0.4 0.6 0.8 1.0

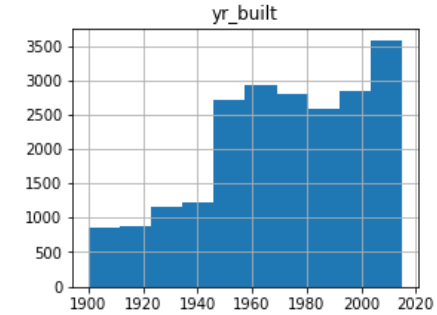

view
0 5000 10000 15000 20000
0 1 2 3 4

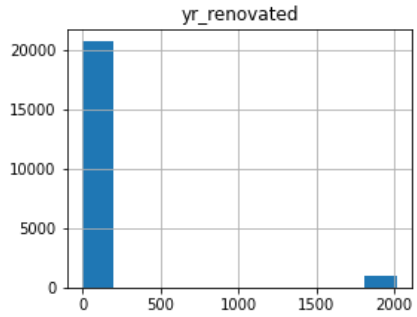

condition
0 2000 4000 6000 8000 10000 12000 14000
1 2 3 4 5

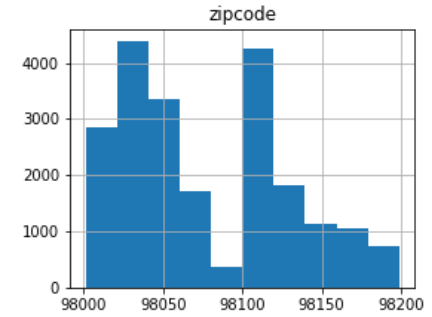

grade
0 2000 4000 6000 8000 10000 12000 14000
2 4 6 8 10 12

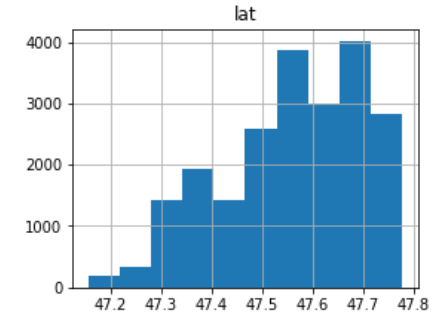

sqft_above
0 2000 4000 6000 8000 10000
0 2000 4000 6000 8000

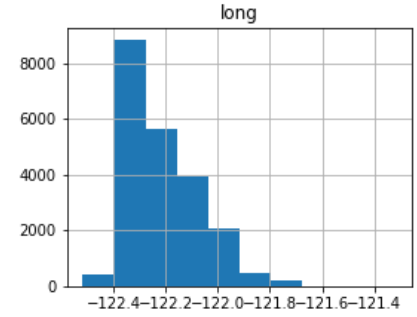

sqft_basement
0 2500 5000 7500 10000 12500 15000
0 1000 2000 3000 4000 5000

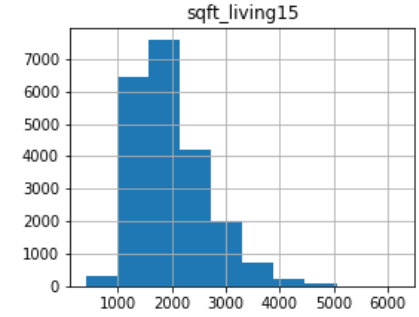

yr_built
0 500 1000 1500 2000 2500 3000 3500
1900 1920 1940 1960 1980 2000 2020

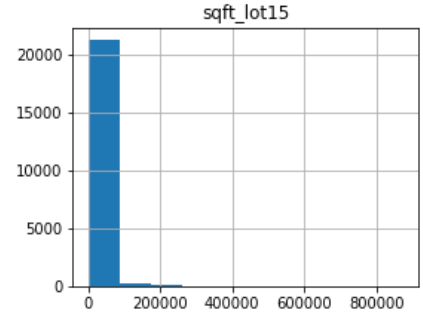

yr_renovated
0 5000 10000 15000 20000
0 500 1000 1500 2000


zipcode
0 1000 2000 3000 4000
98000 98050 98100 98150 98200

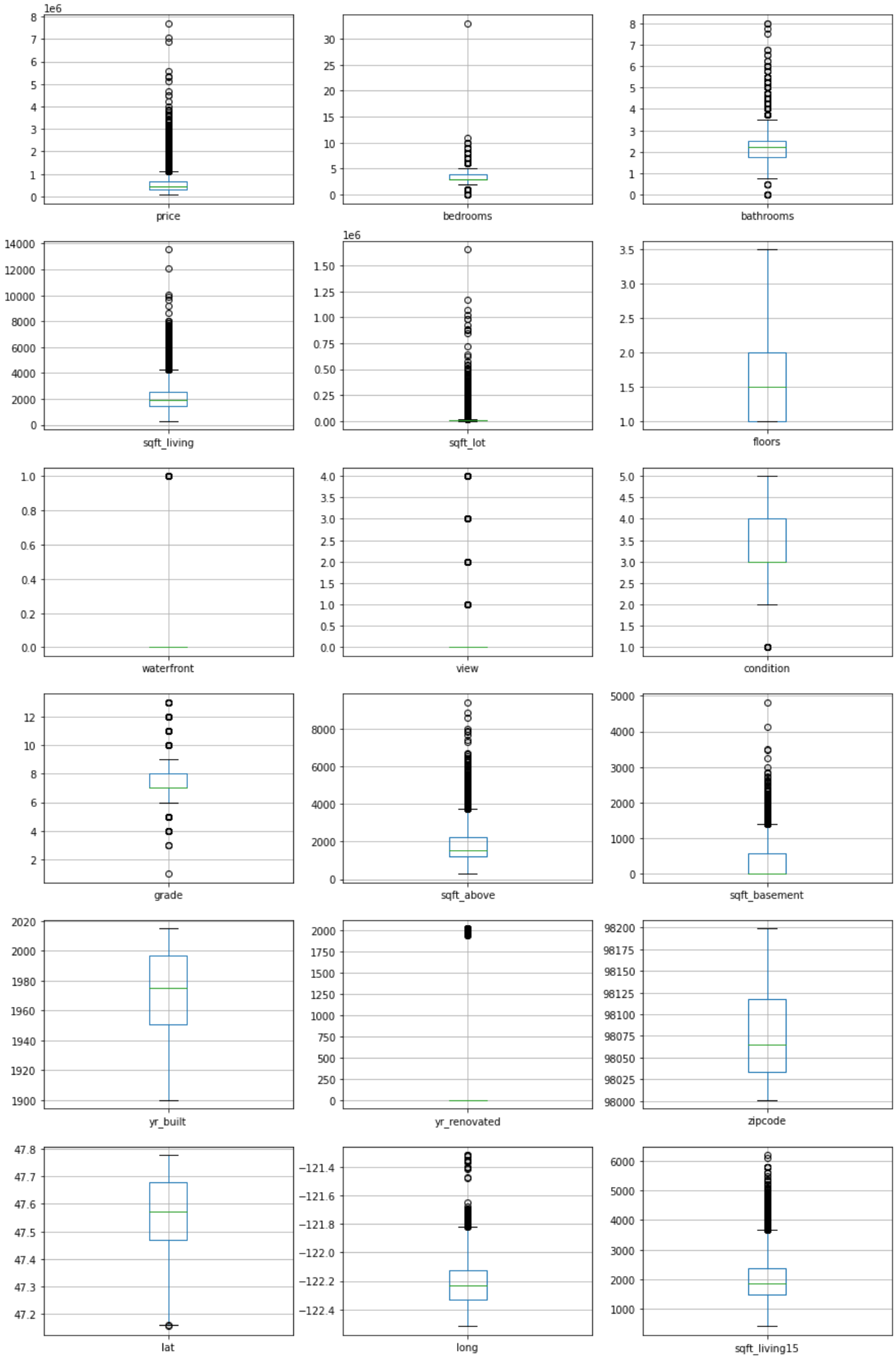

lat
0 1000 2000 3000 4000
47.2 47.3 47.4 47.5 47.6 47.7 47.8

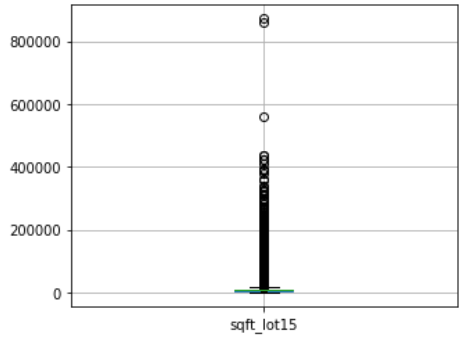

long
0 2000 4000 6000 8000
−122.4 −122.2 −122.0 −121.8 −121.6 −121.4


sqft_living15
0 1000 2000 3000 4000 5000 6000 7000
1000 2000 3000 4000 5000 6000


sqft_lot15
0 5000 10000 15000 20000
0 200000 400000 600000 800000

e. داده های پرت یا outlier ها توسط boxplot به صورت دایره برای همه ی ویژگی ها در نمودارهای زیر مشخص شده اند.





در بعضی از ویژگی ها مانند price, sqft_lot, sqft_living, sqft_above, sqft_basement,... تعداد بسیاری داده ی پرت وجود دارد که در مراحل بعد (پاکسازی داده) از دیتاست حذف خواهند شد.

f. ماتریس عدم شباهت برای دیتاست با استفاده از تابع نوشته شده در کد زیر (فاصله ی اقلیدسی) برای 5 رکورد اول دیتاست به دست آمده.

همچنین می توان آن را برای هر تعداد داده ی دیگر به دست آورد. فقط باید توجه داشت که هرچه تعداد بیشتری داده انتخاب شود، زمان بیشتری برای محاسبه ی ماتریس عدم شباهت صرف میشود.

```
from scipy.spatial.distance import euclidean, pdist, squareform

def similarity_func(u, v):
    return 1 / (1 + euclidean(u,v))
df_num = df[:6].drop(['date'], axis=1)
dists = pdist(df_num, similarity_func)
df_dist = pd.DataFrame(squareform(dists), columns=df_num.index,
                       index=df_num.index)
df_dist
```

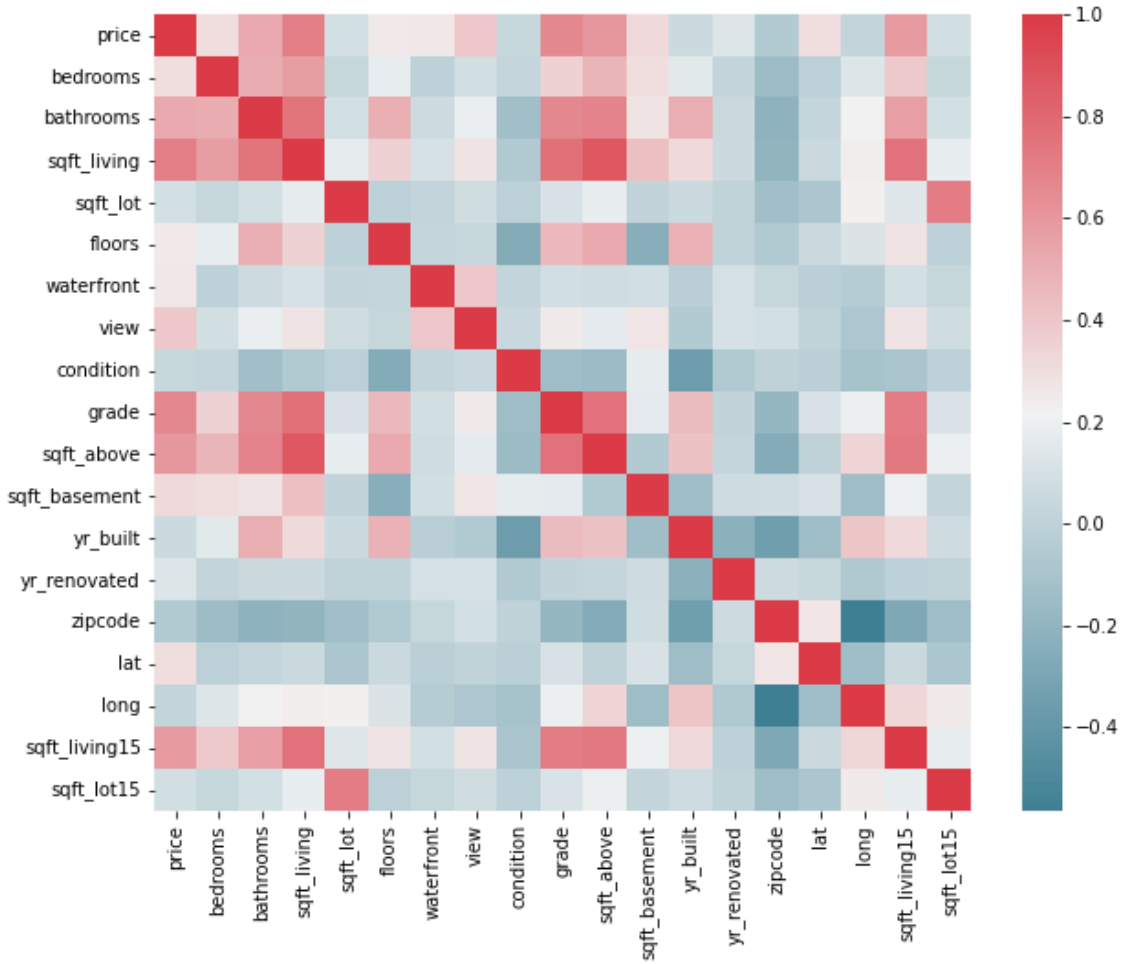
id	7129300520	6414100192	5631500400	2487200875	1954400510	7237550310
id						
7129300520	0.000000e+00	0.000003	2.368433e-05	0.000003	0.000003	9.878305e-07
6414100192	3.163330e-06	0.000000	2.793090e-06	0.000015	0.000036	1.428786e-06
5631500400	2.368433e-05	0.000003	0.000000e+00	0.000002	0.000003	9.494457e-07
2487200875	2.617088e-06	0.000015	2.358232e-06	0.000000	0.000011	1.572387e-06
1954400510	3.470795e-06	0.000036	3.030203e-06	0.000011	0.000000	1.374928e-06
7237550310	9.878305e-07	0.000001	9.494457e-07	0.000002	0.000001	0.000000e+00

g. همبستگی ویژگی ها حساب شده و در شکل زیر به صورت heatmap قابل مشاهده و بررسی میباشد.

همانطور که از شکل برمی آید، هر ویژگی با خودش قویترین همبستگی را دارد.

همچنین مشاهده میشود که در بین ویژگی ها هم همبستگی مثبت و هم منفی وجود دارد که به این معناست که با کاهش یکی دیگری افزایش میابد.

برای مثال میتوان دید که با افزایش سال ساخت، قیمت خانه کاهش می یابد و بالعکس با افزایش مساحت ملک، قیمت نیز همسو با آن افزایش می یابد.

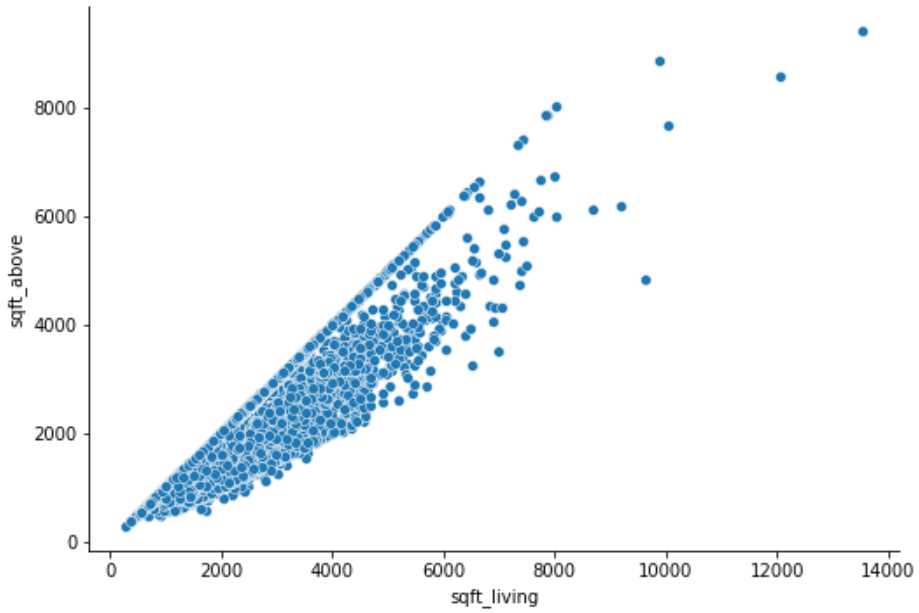


h. با کمک کد زیر ویژگی هایی که همبستگی قوی (چه مثبت و هم جهت و چه منفی و در خلاف جهت) دارند چاپ می شوند و در ادامه نمودار scatter مربوط به آنها رسم می گردد.

```
corr_abs = corr.abs()
up_tri = corr_abs.where(np.triu(np.ones(corr_abs.shape),k=1).astype(np.bool))
col_corr = [column for column in up_tri.columns if any(up_tri[column] > 0.8)]
for col in col_corr:
    for feat in corr:
        if corr[col][feat] > 0.8 and corr[col][feat] < 1:
            print(f'*{col}* and *{feat}* are correlated!')
```

```
*sqft_above* and *sqft_living* are correlated!
```

```
sns.pairplot(df, height=5, aspect=1.5, y_vars=['sqft_above'],
             x_vars=['sqft_living'])
```



همان گونه که انتظار میرفت، نمودار scatter مربوط به این جفت از ویژگی ها نشانگر یک همبستگی قوی است.

2. پیش پردازش

a. برای پاکسازی داده ها ابتدا به ستون date می پردازیم. از آنجا که نوع داده های این ستون از جنس رشته است، باید به نوع datetime تبدیل شود.

در ادامه نیز چک میشود که اگر داده ای با فرمت متفاوتی بود مشخص شود که همانطور که مشخص می باشد چنین اتفاقی در این دیتاست رخ نمی دهد.

```
# fixing the date format
df['date'] = pd.to_datetime(df['date'])

# find out if there is an incorrect date
pd.to_datetime(df['date'], errors='coerce').isnull().value_counts()
```

```
False    21613
Name: date, dtype: int64
```

در ادامه داده های پرت مشخص شده و از دیتاست حذف میشوند. تعداد 2194 داده ی پرت در کل حذف شده است.

در ادامه نیز برای هر ستون تعداد داده ی از دست رفته مشخص میشود که برای همه ی ستون ها، این تعداد برابر 0 است.

```
# remove outliers

def remove_outlier(df):
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)
    IQR = Q3 - Q1
    clean = ~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR)))
    return clean

df_clean = df.copy()
for col in df.columns:
    if col != 'date':
        df_clean = df[remove_outlier(df[col])]

n_outliers = len(df) - len(df_non_out)
print(f'{n_outliers} outliers found!')
print(f'original data had {len(df)} rows')
print(f'cleaned data has {len(df_clean)} rows')
```

```
2194 outliers found!
original data had 21613 rows
cleaned data has 19419 rows
```

```python
for col in df_clean.columns:
    print(f'{df_clean[col].isnull().sum()} missing values in {col}')
```

```
0 missing values in date
0 missing values in price
0 missing values in bedrooms
0 missing values in bathrooms
0 missing values in sqft_living
0 missing values in sqft_lot
0 missing values in floors
0 missing values in waterfront
0 missing values in view
0 missing values in condition
0 missing values in grade
0 missing values in sqft_above
0 missing values in sqft_basement
0 missing values in yr_built
0 missing values in yr_renovated
0 missing values in zipcode
0 missing values in lat
0 missing values in long
0 missing values in sqft_living15
0 missing values in sqft_lot15
```

b. بررسی افزونگی در سطح رکورد نشان میدهد که هیچ رکوردی بیش از یک بار ظاهر نمی شود بنابراین لازم به عملیات کاهش نیست. همینطور برای کاهش افزونگی در سطح ویژگی میتوان ویژگی هایی که مفهوم مشابه دارند را حذف نمود مانند دو ویژگی زیر:

```python
# record redundancy
print(f'{len(df_clean.drop_duplicates()) - len(df_clean)} redundant rows')

# feature redundancy
print('\n*sqft_living15* and *sqft_lot15* can be removed')
```

```
0 redundant rows

*sqft_living15* and *sqft_lot15* can be removed
```

c. همانگونه که بالاتر اشاره شد، بررسی افزونگی در سطح رکورد نشان میدهد که هیچ رکوردی بیش از یک بار ظاهر نمی شود بنابراین لازم به عملیات کاهش نیست.

d. عملیات نرمال سازی توسط تابع نوشته شده برای تمامی ستون ها بجز ستون date اعمال میشود. نمونه ای از حاصل نرمال سازی چند ستون اول در ادامه قابل مشاهده است:

```
df_norm = df_clean.copy()
for col in df_clean.columns:
    if col != 'date':
        df_norm[col] = (df_clean[col]
                        - df_clean[col].mean())
        / (df_clean[col].max() - df_clean[col].min())
```

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
7129300520	2014-10-13	-0.038870	-0.010705	-0.135125	-0.070291	-0.002245	-0.196076
6414100192	2014-12-09	0.002602	-0.010705	0.021125	0.048716	-0.000878	0.203924
5631500400	2015-02-25	-0.044368	-0.041008	-0.135125	-0.105393	0.001491	-0.196076
2487200875	2014-12-09	0.011261	0.019598	0.114875	-0.003510	-0.002804	-0.196076
1954400510	2015-02-18	-0.001072	-0.010705	-0.010125	-0.027482	-0.000158	-0.196076

e. میتوان از ویژگی هایی که همبستگی قوی با هم دارند مانند sqft_above و sqft_living یکی را انتخاب و حذف کرد. همچنین از راه PCA نیز می توان تعداد ویژگی هایی که موجب توضیح دادن بیش از 95٪ دیتاست را مشخص کرد و بقیه ویژگی ها را حذف نمود.

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

scaler = StandardScaler()
X = scaler.fit_transform(df_clean[[col for col in df.columns if col!='date']])
pca = PCA()
X_r = pca.fit_transform(X)
```

```
pca.explained_variance_ratio_.cumsum()
```

```
array([0.29374255, 0.42273545, 0.52482472, 0.59062838, 0.64590506,
       0.69894424, 0.74714496, 0.79063267, 0.82881635, 0.86413399,
       0.8919153 , 0.91613421, 0.93793644, 0.95296042, 0.9674103 ,
       0.98029438, 0.99069548, 1.        , 1.        ])
```

با توجه به آنالیز بالا، تعداد 15 ویژگی اول موجب توضیح بیش از 95٪ دیتا میشوند. بنابراین مجموعه ویژگی های انتخابی میتواند مجموعه ی زیر باشد:

'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade', 'sqft_above',
'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode'