IDEAS - Institute of Data Engineering, Analytics and Science Foundation
Autumn Internship Program 2025
Project Notebook
Topic: Analysing house prices and predicting price based on other factors related to it

Created by: **Koulika Paul**
Designation: **Project Linked Associate Research Engineer in Statistics**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

```
house_data= pd.read_csv('/content/house_price_india.csv')
house_data
```

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Built Year	Renovation Year	Postal Code
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4	5	...	1921	0	122003
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0	5	...	1909	0	122004
2	6762810998	42491	5	2.75	2910	9480	1.5	0	0	3	...	1939	0	122004
3	6762812605	42491	4	2.50	3310	42998	2.0	0	0	3	...	2001	0	122005
4	6762812919	42491	3	2.00	2710	4500	1.5	0	0	4	...	1929	0	122006
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14615	6762830250	42734	2	1.50	1556	20000	1.0	0	0	4	...	1957	0	122066
14616	6762830339	42734	3	2.00	1680	7000	1.5	0	0	4	...	1968	0	122072
14617	6762830618	42734	2	1.00	1070	6120	1.0	0	0	3	...	1962	0	122056
14618	6762830709	42734	4	1.00	1030	6621	1.0	0	0	4	...	1955	0	122042
14619	6762831463	42734	3	1.00	900	4770	1.0	0	0	3	...	1969	2009	122018

14620 rows × 23 columns

Insert synthetic missing value

```
house_data_missing= house_data.copy()
## each column missing values are inserted, 20% sample of each column is been drawn and its index is noted to replace those rows w
for i in house_data_missing.columns:
  house_data_missing[i].loc[house_data_missing[i].sample(frac=0.2).index]= np.nan
house_data_missing
```

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Built Year	Renovation Year	
0	NaN	42491.0	5.0	NaN	3650.0	NaN	2.0	0.0	4.0	5.0	...	NaN	NaN	12
1	6.762811e+09	42491.0	4.0	NaN	NaN	4000.0	NaN	0.0	0.0	5.0	...	1909.0	0.0	12
2	6.762811e+09	42491.0	NaN	2.75	2910.0	9480.0	1.5	0.0	0.0	3.0	...	1939.0	0.0	12
3	6.762813e+09	NaN	4.0	2.50	NaN	NaN	NaN	0.0	0.0	3.0	...	2001.0	0.0	12
4	6.762813e+09	42491.0	3.0	2.00	2710.0	NaN	1.5	0.0	0.0	4.0	...	1929.0	0.0	12
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
14615	6.762830e+09	NaN	2.0	NaN	1556.0	20000.0	1.0	0.0	NaN	4.0	...	1957.0	NaN	12
14616	6.762830e+09	42734.0	3.0	2.00	1680.0	7000.0	NaN	0.0	0.0	4.0	...	1968.0	0.0	12
14617	6.762831e+09	42734.0	2.0	1.00	1070.0	6120.0	1.0	NaN	0.0	3.0	...	1962.0	0.0	12
14618	NaN	42734.0	4.0	1.00	1030.0	6621.0	1.0	NaN	0.0	4.0	...	1955.0	0.0	12
14619	6.762831e+09	NaN	3.0	1.00	NaN	4770.0	NaN	0.0	0.0	3.0	...	1969.0	2009.0	12

14620 rows × 23 columns

Q1. Try inserting a missing value to a specific column of your choice

```
house_data_missing['living area'] = house_data_missing['living area'].fillna(house_data_missing['living area'].mean())
print(house_data_missing['living area'])
# tried for the column living area as given of your choice.
```

```
0        3650.000000
1        2099.477172
2        2910.000000
3        2099.477172
4        2710.000000
            ...
14615    1556.000000
14616    1680.000000
14617    1070.000000
14618    1030.000000
14619    2099.477172
Name: living area, Length: 14620, dtype: float64
```

Information about the data

```
house_data_missing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 16 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   id                                    11696 non-null  float64
 1   number of bedrooms                    11696 non-null  float64
 2   number of bathrooms                   11696 non-null  float64
 3   living area                           14620 non-null  float64
 4   lot area                              11696 non-null  float64
 5   number of floors                      11696 non-null  float64
 6   waterfront present                    11696 non-null  float64
 7   number of views                       11696 non-null  float64
 8   condition of the house                11696 non-null  float64
 9   grade of the house                    11696 non-null  float64
 10  Area of the house(excluding basement)  11696 non-null  float64
 11  Area of the basement                  11696 non-null  float64
 12  Built Year                            11696 non-null  float64
 13  Number of schools nearby              11696 non-null  float64
 14  Distance from the airport             11696 non-null  float64
 15  Price                                 11696 non-null  float64
dtypes: float64(16)
memory usage: 1.8 MB
```

Dropping columns

```
house_data_missing.drop(['Date','Longitude','Renovation Year','Postal Code',
                         'Lattitude','living_area_renov', 'lot_area_renov'],
                        axis=1, inplace=True, errors='ignore')

# Print the updated DataFrame
print(house_data_missing)
```

```
                 id  number of bedrooms  number of bathrooms  living area  \
0               NaN                 5.0                  NaN  3650.000000
1      6.762811e+09                 4.0                  NaN  2099.477172
2      6.762811e+09                 NaN                 2.75  2910.000000
3      6.762813e+09                 4.0                 2.50  2099.477172
4      6.762813e+09                 3.0                 2.00  2710.000000
...             ...                 ...                  ...          ...
14615  6.762830e+09                 2.0                  NaN  1556.000000
14616  6.762830e+09                 3.0                 2.00  1680.000000
14617  6.762831e+09                 2.0                 1.00  1070.000000
14618           NaN                 4.0                 1.00  1030.000000
14619  6.762831e+09                 3.0                 1.00  2099.477172

       lot area  number of floors  waterfront present  number of views  \
0           NaN               2.0                 0.0              4.0
1        4000.0               NaN                 0.0              0.0
2        9480.0               1.5                 0.0              0.0
3           NaN               NaN                 0.0              0.0
4           NaN               1.5                 0.0              0.0
...         ...               ...                 ...              ...
14615   20000.0               1.0                 0.0              NaN
14616    7000.0               NaN                 0.0              0.0
14617    6120.0               1.0                 NaN              0.0
14618    6621.0               1.0                 NaN              0.0
14619    4770.0               NaN                 0.0              0.0

       condition of the house  grade of the house  \
0                         5.0                 NaN
1                         5.0                 8.0
2                         3.0                 NaN
3                         3.0                 9.0
4                         4.0                 8.0
...                       ...                 ...
14615                     4.0                 NaN
14616                     4.0                 7.0
14617                     3.0                 6.0
14618                     4.0                 NaN
14619                     3.0                 6.0

       Area of the house(excluding basement)  Area of the basement  \
0                                        NaN                280.0
1                                     1910.0               1010.0
2                                     2910.0                  0.0
3                                     3310.0                  0.0
4                                     1880.0                830.0
...                                      ...                  ...
14615                                 1556.0                  0.0
14616                                    NaN                  NaN
14617                                    NaN                  0.0
14618                                    NaN                  0.0
14619                                  900.0                  NaN

       Built Year  Number of schools nearby  Distance from the airport  \
0             NaN                       2.0                        NaN
1          1909.0                       2.0                       51.0
2          1939.0                       1.0                       53.0
3          2001.0                       NaN                       76.0
4          1929.0                       NaN                       51.0
```

```
house_data_missing.columns
```

```
Index(['id', 'number of bedrooms', 'number of bathrooms', 'living area',
       'lot area', 'number of floors', 'waterfront present', 'number of views',
       'condition of the house', 'grade of the house',
       'Area of the house(excluding basement)', 'Area of the basement',
       'Built Year', 'Number of schools nearby', 'Distance from the airport',
       'Price'],
      dtype='object')
```

Q2. Show statistics about the data of only numeric columns

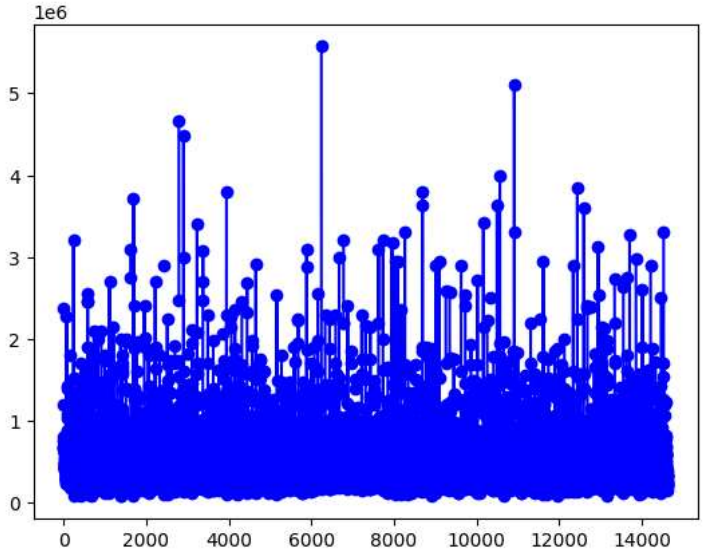
```
print(house_data_missing.select_dtypes(include='number').columns)
# Show statistics (count, mean, std, min, max, quartiles) of numeric columns
```

```
print(house_data_missing.describe())
```

```
count  1.169600e+04        11696.000000         11696.000000  14620.000000
mean   6.762821e+09            3.381669             2.128591   2099.477172
std    6.250445e+03            0.949257             0.773221    829.899436
min    6.762810e+09            1.000000             0.500000    370.000000
25%    6.762815e+09            3.000000             1.750000   1560.000000
50%    6.762821e+09            3.000000             2.250000   2099.477172
75%    6.762826e+09            4.000000             2.500000   2390.000000
max    6.762832e+09           33.000000             8.000000  13540.000000

           lot area  number of floors  waterfront present  number of views  \
count  1.169600e+04      11696.000000        11696.000000     11696.000000
mean   1.512863e+04          1.505130            0.007524         0.237517
std    3.897649e+04          0.539598            0.086417         0.773447
min    5.200000e+02          1.000000            0.000000         0.000000
25%    5.006000e+03          1.000000            0.000000         0.000000
50%    7.620000e+03          1.500000            0.000000         0.000000
75%    1.075200e+04          2.000000            0.000000         0.000000
max    1.074218e+06          3.500000            1.000000         4.000000

       condition of the house  grade of the house  \
count            11696.000000        11696.000000
mean                 3.432199            7.670315
std                  0.663740            1.167199
min                  1.000000            4.000000
25%                  3.000000            7.000000
50%                  3.000000            7.000000
75%                  4.000000            8.000000
max                  5.000000           13.000000

       Area of the house(excluding basement)  Area of the basement  \
count                           11696.000000          11696.000000
mean                             1801.594391            293.110123
std                               829.043165            445.571406
min                               380.000000              0.000000
25%                              1200.000000              0.000000
50%                              1570.000000              0.000000
75%                              2242.000000            570.000000
max                              8860.000000           4820.000000

         Built Year  Number of schools nearby  Distance from the airport  \
count  11696.000000               11696.000000               11696.000000
mean    1971.080626                   2.016245                  64.988458
std       29.461211                   0.818392                   8.915035
min     1900.000000                   1.000000                  50.000000
25%     1951.000000                   1.000000                  57.000000
50%     1975.000000                   2.000000                  65.000000
75%     1997.000000                   3.000000                  73.000000
max     2015.000000                   3.000000                  80.000000

              Price
count  1.169600e+04
mean   5.365408e+05
std    3.604642e+05
min    7.800000e+04
25%    3.200000e+05
50%    4.490000e+05
75%    6.400000e+05
max    5.570000e+06
```

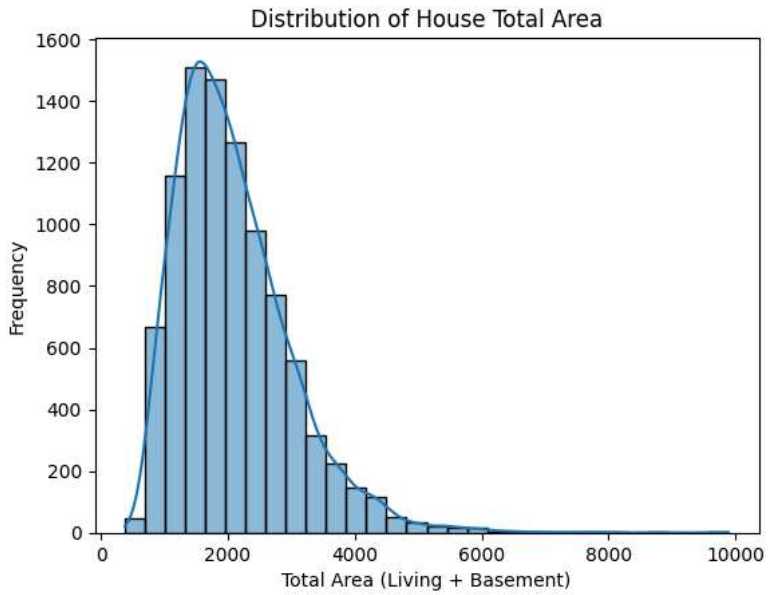
Distribution of price in the dataset

```
plt.plot(house_data_missing.index,house_data_missing['Price'],marker='o',linestyle='-',color='b', label="Distribution of Price for
plt.show()
```



Q3. Find the distribution of area (total area) of houses (use Seaborn distplot)

```
# Create a new column 'total_area'
# as their is no such column total area so for total area(area of house (excluding basement)+ area of basement)

house_data_missing['total_area'] = (
    house_data_missing['Area of the house(excluding basement)'] +
    house_data_missing['Area of the basement']
)
import seaborn as sns
import matplotlib.pyplot as plt

sns.histplot(house_data_missing['total_area'], bins=30, kde=True)

plt.title("Distribution of House Total Area")
plt.xlabel("Total Area (Living + Basement)")
plt.ylabel("Frequency")
plt.show()
```



Checking for duplicate rows

```
house_data_missing.duplicated().sum()
```

```
np.int64(0)
```

Checking missing values

```
house_data_missing.isna().sum()
```

	0
id	2924
number of bedrooms	2924
number of bathrooms	2924
living area	0
lot area	2924
number of floors	2924
waterfront present	2924
number of views	2924
condition of the house	2924
grade of the house	2924
Area of the house(excluding basement)	2924
Area of the basement	2924
Built Year	2924
Number of schools nearby	2924
Distance from the airport	2924
Price	2924
total_area	5228

dtype: int64

Handling missing values

Technique 1: Remove missing value rows

```
house_data_missing1= house_data_missing.dropna()
house_data_missing1.isna().sum()
```

	0
id	0
number of bedrooms	0
number of bathrooms	0
living area	0
lot area	0
number of floors	0
waterfront present	0
number of views	0
condition of the house	0
grade of the house	0
Area of the house(excluding basement)	0
Area of the basement	0
Built Year	0
Number of schools nearby	0
Distance from the airport	0
Price	0
total_area	0

dtype: int64

Technique 2: Replace with mean

```
house_data_missing2= house_data_missing.copy()
for cols in house_data_missing2.columns:
  house_data_missing2[cols]= house_data_missing2[cols].fillna(np.mean(house_data_missing2[cols]))
house_data_missing2.isna().sum()
```

	0
id	0
number of bedrooms	0
number of bathrooms	0
living area	0
lot area	0
number of floors	0
waterfront present	0
number of views	0
condition of the house	0
grade of the house	0
Area of the house(excluding basement)	0
Area of the basement	0
Built Year	0
Number of schools nearby	0
Distance from the airport	0
Price	0
total_area	0

dtype: int64

Q4. Try replacing the missing values with the standard deviation of each column

```
#replace the missing values with the standard deviation  for each column

house_data_missing2 = house_data_missing.copy()

for cols in house_data_missing2.columns:
    house_data_missing2[cols] = house_data_missing2[cols].fillna(house_data_missing2[cols].std())

print(house_data_missing2.isna().sum())
```

```
id                                    0
number of bedrooms                    0
number of bathrooms                   0
living area                           0
lot area                              0
number of floors                      0
waterfront present                    0
number of views                       0
condition of the house                0
grade of the house                    0
Area of the house(excluding basement)    0
Area of the basement                  0
Built Year                            0
Number of schools nearby              0
Distance from the airport             0
Price                                 0
total_area                            0
dtype: int64
```

Technique 3: Interpolation

```
house_data_missing3= house_data_missing.interpolate(method='linear')
house_data_missing2.isna().sum()
```

	0
id	0
number of bedrooms	0
number of bathrooms	0
living area	0
lot area	0
number of floors	0
waterfront present	0
number of views	0
condition of the house	0
grade of the house	0
Area of the house(excluding basement)	0
Area of the basement	0
Built Year	0
Number of schools nearby	0
Distance from the airport	0
Price	0
total_area	0

dtype: int64

Q5. Try replacing the missing values using interpolation with the polynomial method

```
house_data_missing4 = house_data_missing.copy()

# Polynomial interpolation (degree=2 for quadratic)
house_data_missing4 = house_data_missing4.interpolate(method='polynomial', order=2)
```

```
print(house_data_missing4.isna().sum())
```

```
id                                   1
number of bedrooms                   0
number of bathrooms                  2
living area                          0
lot area                             1
number of floors                     1
waterfront present                   0
number of views                      0
condition of the house               0
grade of the house                   1
Area of the house(excluding basement)    1
Area of the basement                 1
Built Year                           1
Number of schools nearby             1
Distance from the airport            1
Price                                0
total_area                           5
dtype: int64
```

Technique 4: KNN imputation

```
from sklearn.impute import KNNImputer
imputed_vals= KNNImputer(n_neighbors=5)
imputed_data= pd.DataFrame(imputed_vals.fit_transform(house_data_missing),columns= house_data_missing.columns)
imputed_data.isna().sum()
```

	0
id	0
number of bedrooms	0
number of bathrooms	0
living area	0
lot area	0
number of floors	0
waterfront present	0
number of views	0
condition of the house	0
grade of the house	0
Area of the house(excluding basement)	0
Area of the basement	0
Built Year	0
Number of schools nearby	0
Distance from the airport	0
Price	0
total_area	0

dtype: int64

Q6. Perform replacing missing values with KNN imputers on scaled data. Also, inverse the scaled data to get the original data.

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.impute import KNNImputer
house_data_imputed = house_data_missing.copy()
numeric_cols = house_data_imputed.select_dtypes(include=['number']).columns

# Initialize scaler and imputer
scaler = MinMaxScaler()
imputer = KNNImputer(n_neighbors=5)

# Scale, impute, and inverse scale
house_data_imputed[numeric_cols] = scaler.inverse_transform(
```

```
    imputer.fit_transform(
        scaler.fit_transform(house_data_imputed[numeric_cols])
    )
)

print(house_data_imputed.isna().sum())
```

```
id                                    0
number of bedrooms                    0
number of bathrooms                   0
living area                           0
lot area                              0
number of floors                      0
waterfront present                    0
number of views                       0
condition of the house                0
grade of the house                    0
Area of the house(excluding basement)    0
Area of the basement                  0
Built Year                            0
Number of schools nearby              0
Distance from the airport             0
Price                                 0
total_area                            0
dtype: int64
```

Changing datatype

```
imputed_data['Number of schools nearby']=house_data_missing['Number of schools nearby'].astype('int')
```

Find the number of houses having 3 schools nearby

```
imputed_data.groupby('Number of schools nearby')['id'].count()
```

Number of schools nearby	id
1.0	3833
1.2	37
1.4	120
1.6	311
1.8	476
2.0	4405
2.2	546
2.4	402
2.6	325
2.8	142
3.0	4023

dtype: int64

Q7. Interpret the above result

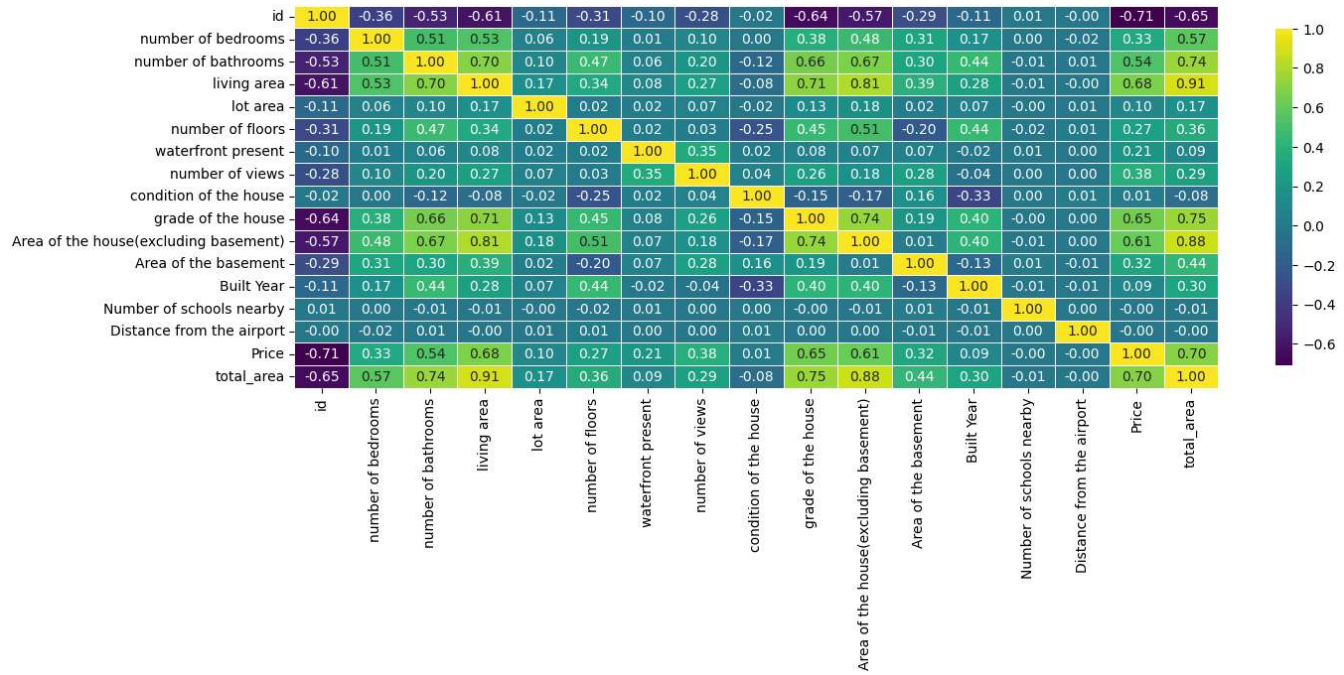
```
imputed_data.groupby('Number of schools nearby')['id'].count()

1.Majority of houses are near 1-2 schools, which might be considered convenient for families.
2.The number of schools nearby can influence property value, demand, or desirability.
3.This grouped count helps in understanding how well-distributed houses are with respect to educational facilities.
4.Very few houses have no schools nearby, which could affect attractiveness or value.
```

Number of schools nearby	id
1.0	3833
1.2	37
1.4	120
1.6	311
1.8	476
2.0	4405
2.2	546
2.4	402
2.6	325
2.8	142
3.0	4023

dtype: int64

Correlation between all the features

```
import seaborn as sns

plt.figure(figsize=(15,5))
sns.heatmap(imputed_data.corr(),annot=True,fmt='0.2f',cmap='viridis',linewidths=0.5,
            cbar_kws={"shrink": 0.88})
plt.show()
```

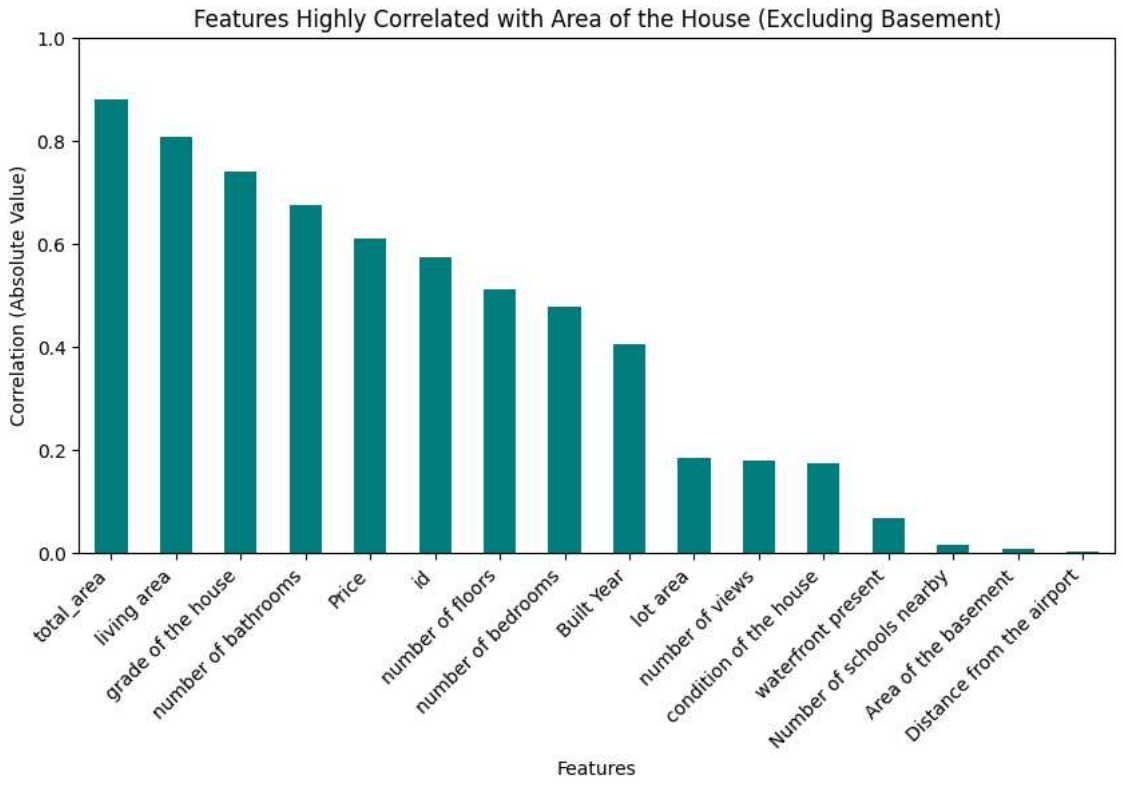


Q8. Find features that are highly correlated with the area of the house (excluding the basement)

```
corr_matrix = imputed_data.corr()
area_corr = corr_matrix['Area of the house(excluding basement)']

# Filter features with high correlation (absolute value >= 0.5), excluding itself
high_corr_features = area_corr[abs(area_corr) >= 0.5].drop('Area of the house(excluding basement)')

print("Features highly correlated with Area of the house (excluding basement):")
print(high_corr_features.sort_values(ascending=False))
```

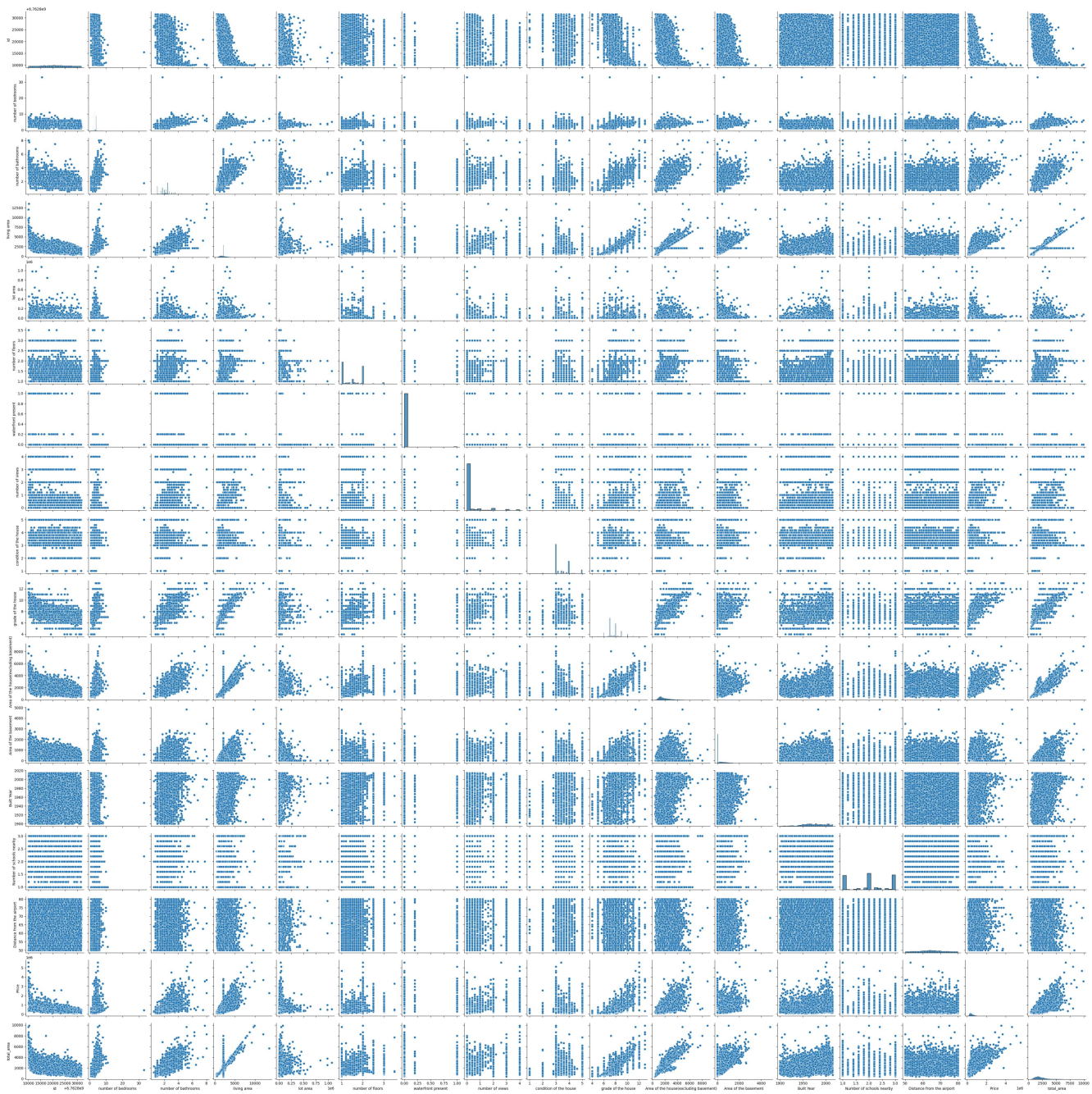
```
Features highly correlated with Area of the house (excluding basement):
total_area             0.879192
living area            0.805690
grade of the house     0.738687
number of bathrooms    0.674675
Price                  0.610475
number of floors       0.509850
id                    -0.574486
Name: Area of the house(excluding basement), dtype: float64
```

```
#visualization of highly correlated with the area of the house (excluding the basement)
import matplotlib.pyplot as plt
corr_matrix = imputed_data.corr()
area_corr = corr_matrix['Area of the house(excluding basement)'].drop('Area of the house(excluding basement)')
area_corr_abs = area_corr.abs().sort_values(ascending=False)

# Plot
plt.figure(figsize=(10,5))
area_corr_abs.plot(kind='bar', color='teal')
plt.title("Features Highly Correlated with Area of the House (Excluding Basement)")
plt.ylabel("Correlation (Absolute Value)")
plt.xlabel("Features")
plt.xticks(rotation=45, ha='right')
plt.ylim(0,1)
plt.show()
```



Find the distribution of each feature

```
sns.pairplot(imputed_data)
plt.show()
```



Q9. Find the average price of houses having 3 schools nearby

```
imputed_data.groupby('Number of schools nearby')['Price'].agg(np.mean)
## OR
np.mean(imputed_data[imputed_data['Number of schools nearby']==3]['Price'])
```

```
np.float64(536904.8716877953)
```

Q10. Find the average area of houses having 5 bedrooms

```
avg_area_by_bedrooms = imputed_data.groupby('number of bedrooms')['Area of the house(excluding basement)'].mean()
print("Average area of houses with 5 bedrooms:", avg_area_by_bedrooms[5])
```

```
Average area of houses with 5 bedrooms: 2448.507019562716
```

Selecting the features for predicting

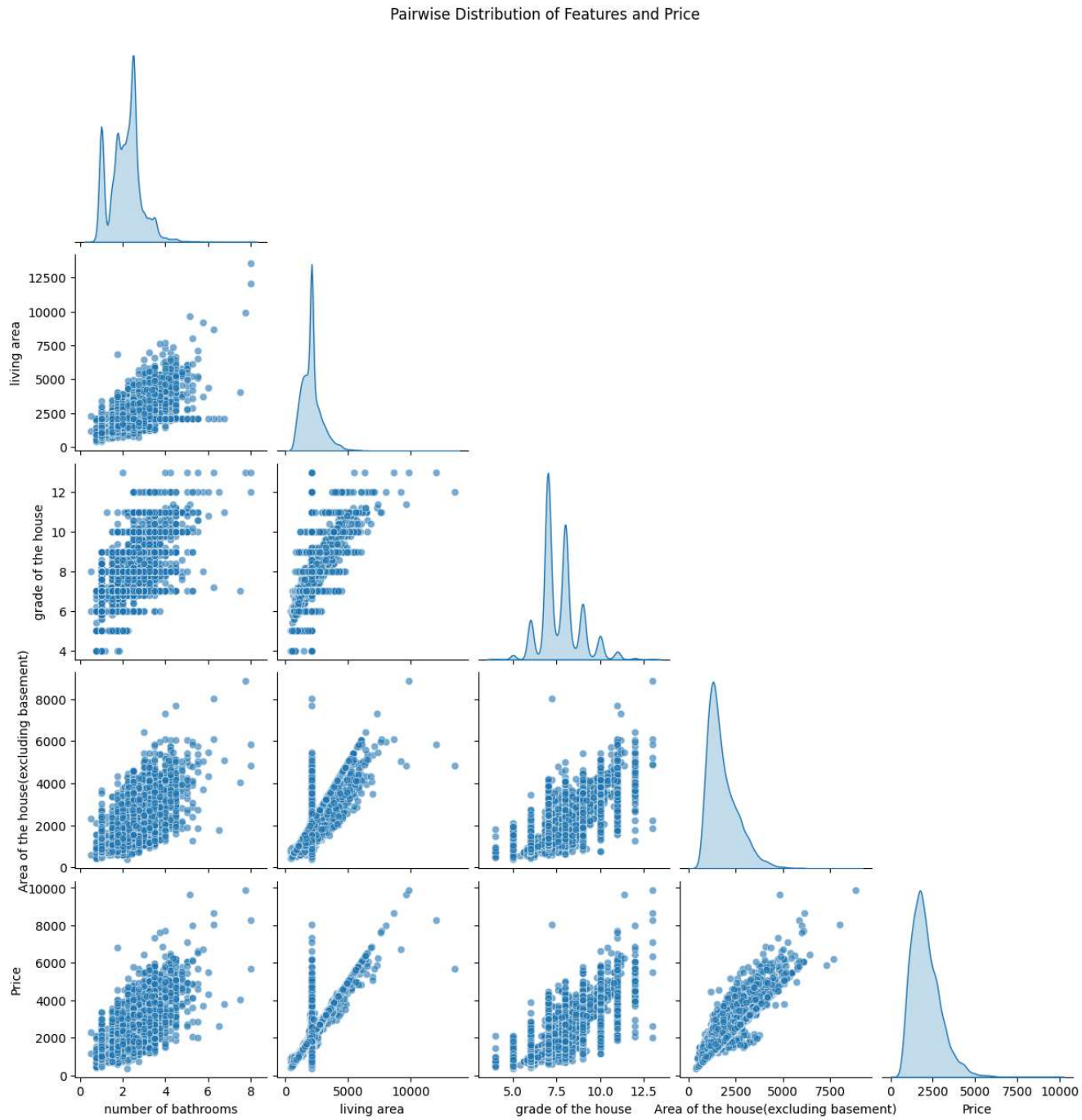
```
features=[]
for cols in imputed_data.iloc[:,:-1].columns:
  if (imputed_data['Price'].corr(imputed_data[cols]))>0.50:
    features.append(cols)
print(features)
```

```
['number of bathrooms', 'living area', 'grade of the house', 'Area of the house(excluding basement)', 'Price']
```

```
X= imputed_data[features]
y= imputed_data.iloc[:,-1]
```

Q11. Show the pairwise distribution of X and y

```
import seaborn as sns
import matplotlib.pyplot as plt
data_for_plot = X.copy()
data_for_plot['Price'] = y
sns.pairplot(data_for_plot, diag_kind='kde', corner=True, plot_kws={'alpha':0.6})
plt.suptitle("Pairwise Distribution of Features and Price", y=1.02)
plt.show()
```



Splitting datasets into training and testing

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.3,random_state=123)
```

Q12. Split the data as 60% training and 40% testing

```
# Split data (60% train, 40% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.4,
    random_state=42
)

print("Training set size:", X_train.shape[0])
print("Testing set size:", X_test.shape[0])
```

```
Training set size: 8772
Testing set size: 5848
```

Fitting the linear regression model and predicting

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,accuracy_score,r2_score
```

```
LR= LinearRegression()
LR_model= LR.fit(X_train,y_train)
y_pred= LR_model.predict(X_test)
```

Evaluation

```
MSE= mean_squared_error(y_test,y_pred)
R_square= r2_score(y_test,y_pred)
```

```
print(MSE)
print("==============")
print(R_square)
```

```
81513.08224518228
==============
0.8932742104537502
```

Q13. Try model building and predicting with some other dataset of your choice

```
# so i am making a linear regression model using salary dataset for predicting the salary
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
df = pd.read_csv("/content/Salary_dataset.csv")
print(df.head())
```

```
   Unnamed: 0  YearsExperience   Salary
0           0              1.2  39344.0
1           1              1.4  46206.0
2           2              1.6  37732.0
3           3              2.1  43526.0
4           4              2.3  39892.0
```

```
X = df[['YearsExperience']]  # Independent variable
y = df['Salary']             # Dependent variable
```