House Prediction Using Backward Elimination

In [2]:
```
# importing libraries

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

In [3]:
```
# importing dataset

dataset=pd.read_csv(r"C:\Users\Jan Saida\OneDrive\Documents\Desktop\Excel sheets\House_data.csv")
dataset
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_abo
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	11
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	21
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	7
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	10
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	16
...	...	...	...	...	...	...	...	...	...	...	...	...	
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.0	0	0	...	8	15
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.0	0	0	...	8	23
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.0	0	0	...	7	10
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.0	0	0	...	8	16
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.0	0	0	...	7	10

21613 rows × 21 columns

In [4]:

```
# checking for missing values

print(dataset.isnull().any())
```

```
id               False
date             False
price            False
bedrooms         False
bathrooms        False
sqft_living      False
sqft_lot         False
floors           False
waterfront       False
view             False
condition        False
grade            False
sqft_above       False
sqft_basement    False
yr_built         False
yr_renovated     False
zipcode          False
lat              False
long             False
sqft_living15    False
sqft_lot15       False
dtype: bool
```

In [5]:

```
# checking for categorical values

print(dataset.dtypes)
```

```
id                 int64
date              object
price            float64
bedrooms           int64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
grade              int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
zipcode            int64
lat              float64
long             float64
sqft_living15      int64
sqft_lot15         int64
dtype: object
```

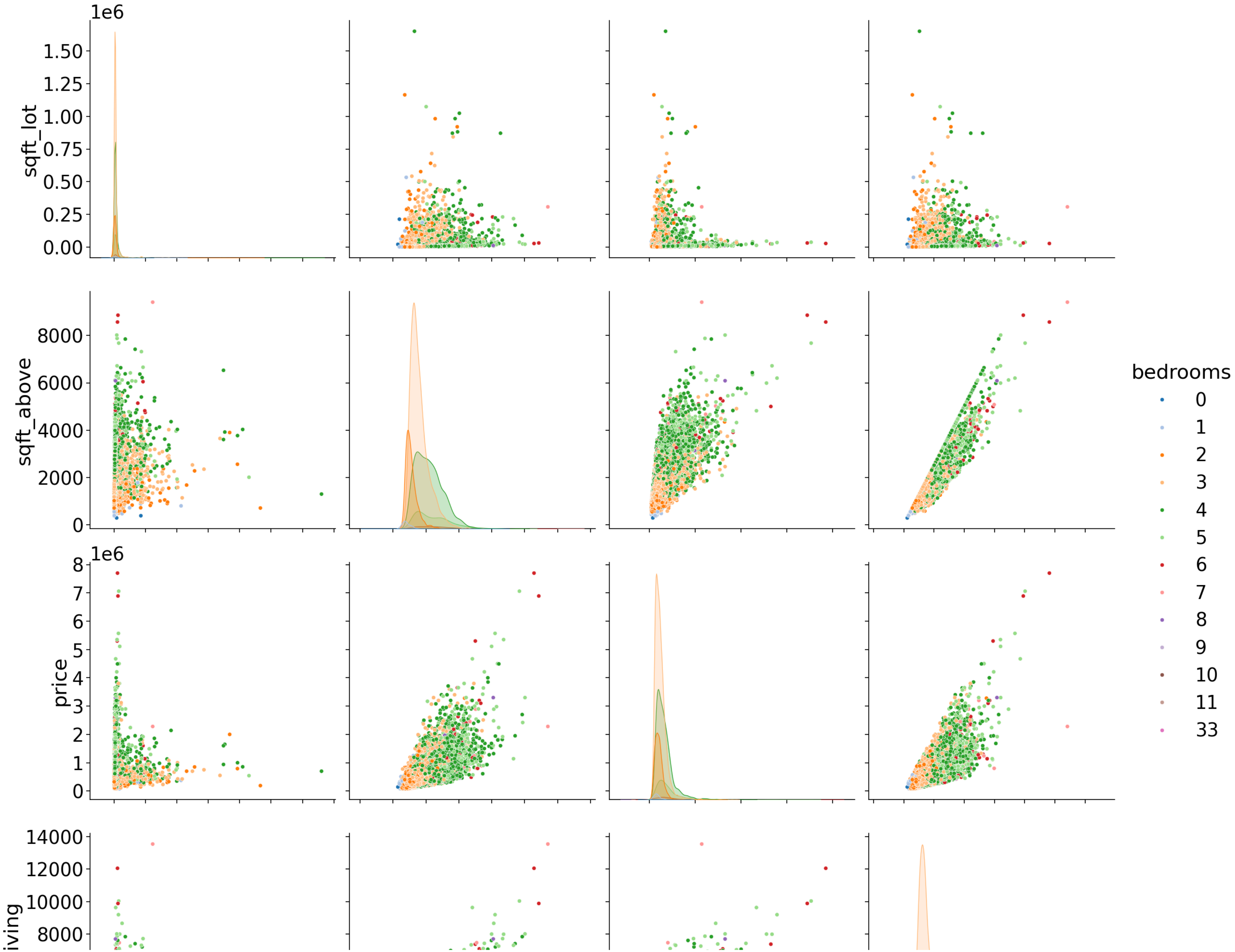
In [6]:
```
# dropping the id and date column

dataset=dataset.drop(['id','date'], axis = 1)
```

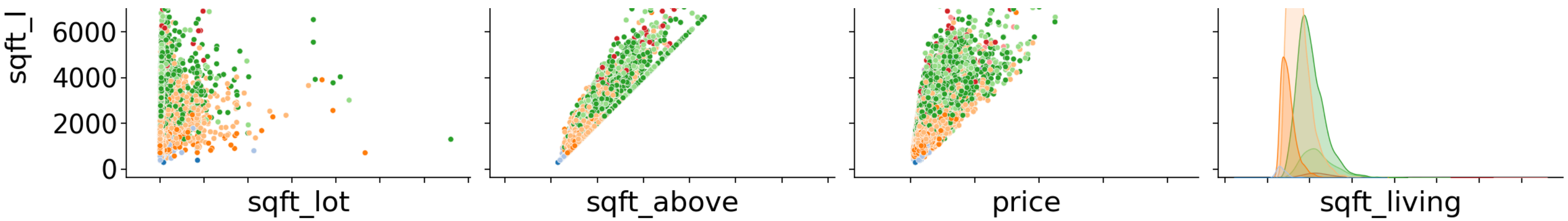
In [7]:
```
# understanding the distribution with seaborn

with sns.plotting_context('notebook',font_scale=2.5):
    g=sns.pairplot(dataset[['sqft_lot','sqft_above','price','sqft_living','bedrooms']],
                   hue='bedrooms',palette='tab20',size=6)
g.set(xticklabels=[]);
```

```
C:\Users\Jan Saida\anaconda3\Lib\site-packages\seaborn\axisgrid.py:2100: UserWarning: The `size` parameter has been renamed to
`height`; please update your code.
  warnings.warn(msg, UserWarning)
```



1e6
sqft_lot
sqft_above
1e6
price
iving
bedrooms
0
1
2
3
4
5
6
7
8
9
10
11
33



In [8]:
```
# separating independent and dependent variable

x=dataset.iloc[:,1:].values
y=dataset.iloc[:,0].values

# splitting dataset into training and testing dataset

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=1/3,random_state=0)
```

In [9]:
```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)

# predicting the test set results

y_pred=regressor.predict(x_test)
```

In [10]:
```
x_train
```

Out[10]:
```
array([[ 3.00000e+00,  1.50000e+00,  1.26000e+03, ..., -1.22123e+02,
         1.80000e+03,  1.03500e+04],
       [ 2.00000e+00,  1.00000e+00,  1.32000e+03, ..., -1.22380e+02,
         1.36000e+03,  2.87300e+03],
       [ 3.00000e+00,  1.00000e+00,  9.20000e+02, ..., -1.22269e+02,
         1.17000e+03,  9.60000e+03],
       ...,
       [ 3.00000e+00,  2.25000e+00,  2.36000e+03, ..., -1.22158e+02,
         2.72000e+03,  1.43880e+04],
       [ 4.00000e+00,  2.00000e+00,  2.37000e+03, ..., -1.22279e+02,
         2.11000e+03,  1.93340e+04],
       [ 4.00000e+00,  2.25000e+00,  2.38000e+03, ..., -1.22120e+02,
         2.23000e+03,  8.92500e+03]])
```

In [11]:
```
x_test
```

Out[11]:
```
array([[ 2.00000e+00,  1.50000e+00,  1.43000e+03, ..., -1.22290e+02,
         1.43000e+03,  1.65000e+03],
       [ 4.00000e+00,  3.25000e+00,  4.67000e+03, ..., -1.22164e+02,
         4.23000e+03,  4.10750e+04],
       [ 2.00000e+00,  7.50000e-01,  1.44000e+03, ..., -1.22364e+02,
         1.44000e+03,  4.30000e+03],
       ...,
       [ 2.00000e+00,  2.00000e+00,  1.87000e+03, ..., -1.22015e+02,
         2.17000e+03,  5.58000e+03],
       [ 2.00000e+00,  1.50000e+00,  1.16000e+03, ..., -1.22315e+02,
         1.16000e+03,  1.00800e+03],
       [ 2.00000e+00,  1.00000e+00,  1.04000e+03, ..., -1.22378e+02,
         1.93000e+03,  5.15000e+03]])
```

In [12]:
```
y_train
```

Out[12]:
```
array([465750., 575000., 212500., ..., 431000., 411000., 699900.])
```

In [13]:
```
y_test
```

Out[13]:
```
array([ 297000., 1580000.,  562100., ...,  592500.,  284900.,  380000.])
```

In [14]:
```
y_pred
```

Out[14]:
```
array([ 386475.41658376, 1517728.88892651,  538760.2916072 , ...,
        526021.14008102,  313813.29875238,  400589.89510017])
```

In [15]:
```python
# backward elimination

import statsmodels.api as sm

def backwardElimination(x, y, SL):
    numVars = len(x[0])
    temp = np.zeros((x.shape[0], numVars))  # Adjusted to use correct shape
    for i in range(0, numVars):
        regressor_OLS = sm.OLS(y, x).fit()
        maxVar = max(regressor_OLS.pvalues)  # Get maximum p-value
        adjR_before = regressor_OLS.rsquared_adj  # Adjusted R-squared before

        if maxVar > SL:
            for j in range(0, numVars - i):
                if regressor_OLS.pvalues[j] == maxVar:  # Find column with max p-value
                    temp[:, j] = x[:, j]
                    x = np.delete(x, j, 1)  # Remove the feature with the max p-value
                    tmp_regressor = sm.OLS(y, x).fit()
                    adjR_after = tmp_regressor.rsquared_adj  # Adjusted R-squared after

                    if adjR_before >= adjR_after:  # If R-squared doesn't improve
                        x_rollback = np.hstack((x, temp[:, [j]]))  # Rollback the deletion
                        return x_rollback  # Return the feature set after rollback
                    else:
                        break  # Continue to the next iteration
        else:
            break  # Stop if the max p-value is less than the significance level

    print(regressor_OLS.summary())
    return x

# Example usage
SL = 0.05
x_opt = x[:, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]]  # Example feature set
x_Modeled = backwardElimination(x_opt, y, SL)
```

In [16]:
```python
x_Modeled
```

Out[16]:
```
array([[3.000e+00, 1.000e+00, 1.180e+03, ..., 1.340e+03, 5.650e+03,
        1.000e+00],
       [3.000e+00, 2.250e+00, 2.570e+03, ..., 1.690e+03, 7.639e+03,
        2.000e+00],
       [2.000e+00, 1.000e+00, 7.700e+02, ..., 2.720e+03, 8.062e+03,
        1.000e+00],
       ...,
       [2.000e+00, 7.500e-01, 1.020e+03, ..., 1.020e+03, 2.007e+03,
        2.000e+00],
       [3.000e+00, 2.500e+00, 1.600e+03, ..., 1.410e+03, 1.287e+03,
        2.000e+00],
       [2.000e+00, 7.500e-01, 1.020e+03, ..., 1.020e+03, 1.357e+03,
        2.000e+00]])
```

In [17]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split

# Assuming you have 'x' and 'y' (your feature matrix and target vector)

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

# Create and train the model
model = LinearRegression()
model.fit(x_train, y_train)  # Train the model on the training data

# Make predictions on the test set
y_pred = model.predict(x_test)

# Calculate performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the performance metrics
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 36326416754.036
R-squared: 0.6949536715546621
```