House Prices using Backward Elimination

Just started with machine learning. I have used backward Elimination to check the usefulness of dependent variables.

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

%matplotlib inline

#importing dataset using pandas
dataset=pd.read_csv(r"D:\Data Science with AI\multiple linear regression\MLR\H

#to see my dataset is comprised of
dataset.head()
```

Out[1]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	w
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	

5 rows × 21 columns

In [2]:
```
#checking if any value is missing
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 [3]:
```
dataset.head()
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	w
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	

5 rows × 21 columns

In [4]:
```
dataset.columns
```

Out[4]:
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

In [5]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64  
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64  
 9   view           21613 non-null  int64  
 10  condition      21613 non-null  int64  
 11  grade          21613 non-null  int64  
 12  sqft_above     21613 non-null  int64  
 13  sqft_basement  21613 non-null  int64  
 14  yr_built       21613 non-null  int64  
 15  yr_renovated   21613 non-null  int64  
 16  zipcode        21613 non-null  int64  
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

In [6]:
```
#checking for categorical data
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 [7]:
```
# Check the actual column names in your DataFrame
print(dataset.columns)
```

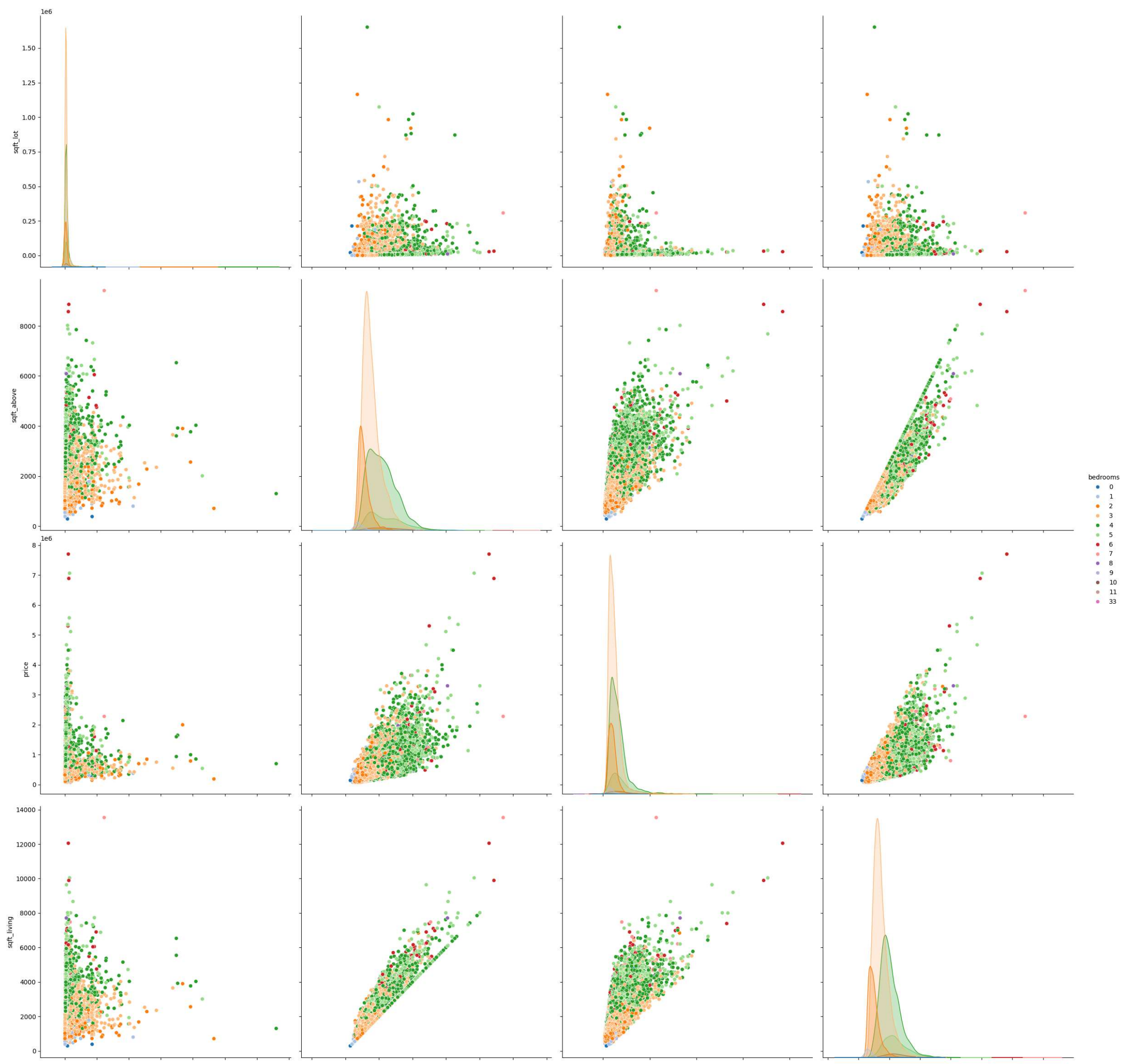
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

In [8]:
```
#droping the id and date columns
dataset = dataset.drop(['id' ,'date'], axis=1)
```

In [12]:
```
# understanding the distribution with seaborn

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

```
C:\Users\Achal Raghorte\AppData\Roaming\Python\Python311\site-packages\seabor
n\axisgrid.py:2100: UserWarning: The `size` parameter has been renamed to `he
ight`; please update your code.
  warnings.warn(msg, UserWarning)
```



In [13]:
```
#seprating independent and dependent variable
x= dataset.iloc[:,1:].values
y = dataset.iloc[:,0].values
```

In [14]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state
```

In [15]:
```
from sklearn.linear_model import LinearRegression
regressor =LinearRegression()   #linar model= relationship between the input f
                  # LinearRegression=for modeling the relationship between a de
regressor.fit(x_train,y_train)
```

Out[15]: LinearRegression()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [16]:
```
#predicting the test set results
y_pred =regressor.predict(x_test)
```

In [17]:
```
#backward elimination
import statsmodels.api as sm
```

In [20]:
```
def backwardElimination(x, SL):
    numVars = len(x[0])
    temp = np.zeros((21613,19)).astype(int)
    for i in range(0, numVars):
        regressor_OLS = sm.OLS(y, x).fit()
        maxVar = max(regressor_OLS.pvalues).astype(float)
        adjR_before = regressor_OLS.rsquared_adj.astype(float)
        if maxVar > SL:
            for j in range(0, numVars - i):
                if (regressor_OLS.pvalues[j].astype(float) == maxVar):
                    temp[:,j] = x[:, j]
                    x = np.delete(x, j, 1)
                    tmp_regressor = sm.OLS(y, x).fit()
                    adjR_after = tmp_regressor.rsquared_adj.astype(float)
                    if (adjR_before >= adjR_after):
                        x_rollback = np.hstack((x, temp[:,[0,j]]))
                        x_rollback = np.delete(x_rollback, j, 1)
                        print (regressor_OLS.summary())
                        return x_rollback
                    else:
                        continue
    regressor_OLS.summary()
    return x

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

```
                                 OLS Regression Results
==============================================================================
==========
Dep. Variable:                      y   R-squared (uncentered):
0.905
Model:                            OLS   Adj. R-squared (uncentered):
0.905
Method:                 Least Squares   F-statistic:
1.211e+04
Date:                Wed, 06 Mar 2024   Prob (F-statistic):
0.00
Time:                        15:52:28   Log-Likelihood:                     -
2.9461e+05
No. Observations:               21613   AIC:
5.892e+05
Df Residuals:                   21596   BIC:
5.894e+05
Df Model:                          17
Covariance Type:            nonrobust
==============================================================================
=
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
-
x1         -3.551e+04   1888.716    -18.802      0.000   -3.92e+04   -3.18e+0
4
x2          4.105e+04   3253.759     12.618      0.000    3.47e+04    4.74e+0
4
x3           110.2642      2.268     48.607      0.000     105.818     114.71
1
x4             0.1334      0.048      2.786      0.005       0.040       0.22
7
x5          5261.5471   3541.347      1.486      0.137   -1679.755    1.22e+0
4
x6          5.833e+05   1.74e+04     33.598      0.000    5.49e+05    6.17e+0
5
x7          5.236e+04   2128.298     24.600      0.000    4.82e+04    5.65e+0
4
x8          2.721e+04   2323.818     11.709      0.000    2.27e+04    3.18e+0
4
x9          9.548e+04   2145.492     44.503      0.000    9.13e+04    9.97e+0
4
x10           71.3928      2.238     31.902      0.000      67.006      75.77
9
x11           38.8714      2.624     14.813      0.000      33.728      44.01
5
x12        -2561.7953     68.006    -37.670      0.000   -2695.092   -2428.49
8
x13           20.4187      3.646      5.600      0.000      13.272      27.56
6
x14         -519.0756     17.826    -29.119      0.000    -554.016    -484.13
6
x15         6.022e+05   1.07e+04     56.106      0.000    5.81e+05    6.23e+0
5
x16        -2.179e+05   1.31e+04    -16.683      0.000   -2.44e+05   -1.92e+0
5
```

```
x17            23.0994      3.392      6.811      0.000      16.452      29.74
7
x18            -0.3761      0.073     -5.137      0.000      -0.520      -0.23
3
==============================================================================
=
Omnibus:                    18403.146   Durbin-Watson:                   1.99
1
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1873534.49
8
Skew:                           3.572   Prob(JB):                         0.0
0
Kurtosis:                      48.049   Cond. No.                     1.82e+1
7
==============================================================================
=

Notes:
[1] R² is computed without centering (uncentered) since the model does not co
ntain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
[3] The smallest eigenvalue is 6.63e-21. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [ ]: