Regresi

https://www.kaggle.com/harlfoxem/housesalespredictio

```
from google.colab import drive

drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
import pandas as pd

df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/kc_house_data.csv")
df
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	3
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	3

```
#checking if any value is missing
print(df.isnull().any())
```

Saved successfully!

```
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
```

```
#checking for categorical data
print(df.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
```



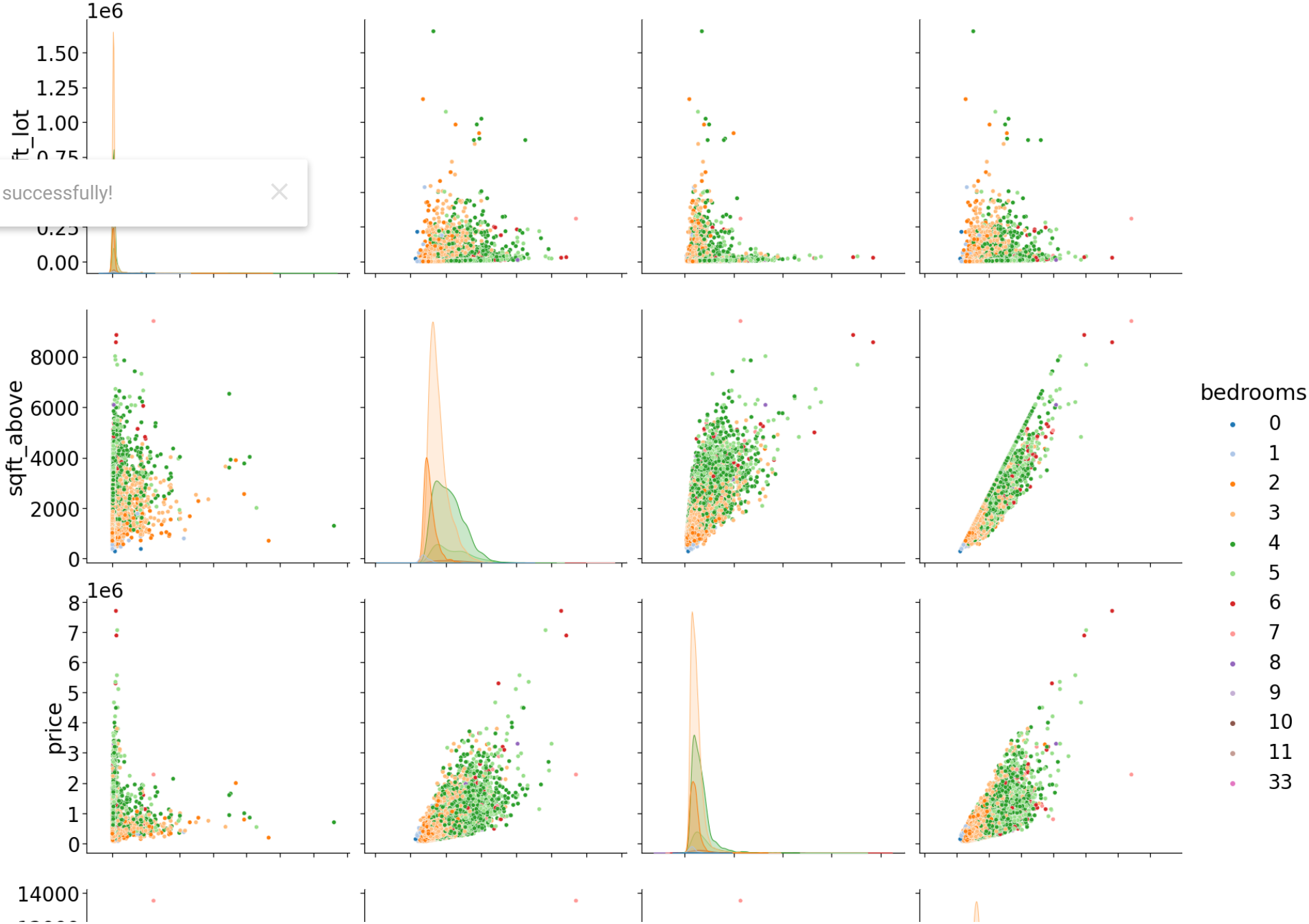
```
zipcode            int64
lat              float64
long             float64
sqft_living15      int64
sqft_lot15         int64
dtype: object
```

```
#dropping the id and date column
dataset = df.drop(['id','date'], axis = 1)
```

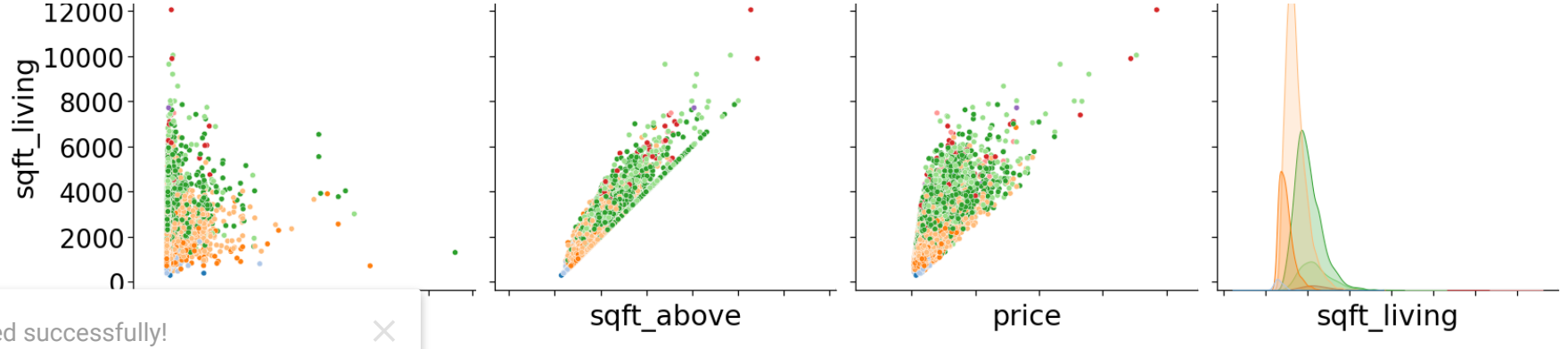
```
import seaborn as sns

#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=[]);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:2076: UserWarning: The `size` parameter has been renamed to `height`
  warnings.warn(msg, UserWarning)
```



Saved successfully!



Saved successfully!

Split Data

```
#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)
```

Modelling

Saved successfully!

```
earRegression

regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)


#Backward Elimination
import statsmodels.api as sm
import numpy as np

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
```

Saved successfully!

```
3,9,10,11,12,13,14,15,16,17]]
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:	Sun, 28 Nov 2021	Prob (F-statistic):	0.00
Time:	01:04:40	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	0.975]
x1	-3.551e+04	1888.716	-18.802	0.000	-3.92e+04	-3.18e+04
x2	4.105e+04	3253.759	12.618	0.000	3.47e+04	4.74e+04
x3	110.2642	2.268	48.607	0.000	105.818	114.711
x4	0.1334	0.048	2.786	0.005	0.040	0.227
x5	5261.5471	3541.347	1.486	0.137	-1679.755	1.22e+04
x6	5.833e+05	1.74e+04	33.598	0.000	5.49e+05	6.17e+05
x7	5.236e+04	2128.298	24.600	0.000	4.82e+04	5.65e+04
x8	2.721e+04	2323.818	11.709	0.000	2.27e+04	3.18e+04
x9	9.548e+04	2145.492	44.503	0.000	9.13e+04	9.97e+04
x10	71.3928	2.238	31.902	0.000	67.006	75.779
x11	38.8714	2.624	14.813	0.000	33.728	44.015
x12	-2561.7953	68.006	-37.670	0.000	-2695.092	-2428.498
x13	20.4187	3.646	5.600	0.000	13.272	27.566

```
x14         -519.0756     17.826    -29.119      0.000    -554.016    -484.136
x15          6.022e+05   1.07e+04     56.106      0.000    5.81e+05    6.23e+05
x16         -2.179e+05   1.31e+04    -16.683      0.000   -2.44e+05   -1.92e+05
x17            23.0994      3.392      6.811      0.000      16.452      29.747
x18            -0.3761      0.073     -5.137      0.000      -0.520      -0.233
==============================================================================
Omnibus:                    18403.146   Durbin-Watson:                   1.991
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1873534.498
Skew:                           3.572   Prob(JB):                         0.00
                               48.049   Cond. No.                     4.93e+17
==============================================================================
```

Saved successfully!

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 9.03e-22. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

```
y_pred
```

```
array([ 386540.99847843, 1516969.01534058,  538662.72575266, ...,
        526000.75505753,  313924.63663331,  400525.6731457 ])
```

```
df = df['price']
```

```
df
```

```
0        221900.0
1        538000.0
2        180000.0
3        604000.0
4        510000.0
           ...
21608    360000.0
21609    400000.0
21610    402101.0
21611    400000.0
```

```
21612    325000.0
Name: price, Length: 21613, dtype: float64
```

```
df['y_pred'] = y_pred
df
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
an IPython kernel.
                                        221900
1                                       538000
2                                       180000
3                                       604000
4                                       510000
                                ...
21609                                   400000
21610                                   402101
21611                                   400000
21612                                   325000
y_pred    [386540.9984784337, 1516969.0153405832, 538662...
Name: price, Length: 21614, dtype: object
```

Saved successfully!