Multiple Linear Regression with sklearn - Exercise Solution

You are given a real estate dataset.

Real estate is one of those examples that every regression course goes through as it is extremely easy to understand and there is a (almost always) certain causal relationship to be found.

The data is located in the file: 'real_estate_price_size_year.csv'.

You are expected to create a multiple linear regression (similar to the one in the lecture), using the new data.

Apart from that, please:

- Display the intercept and coefficient(s)
- Find the R-squared and Adjusted R-squared
- Compare the R-squared and the Adjusted R-squared
- Compare the R-squared of this regression and the simple linear regression where only 'size' was used
- Using the model make a prediction about an apartment with size 750 sq.ft. from 2009
- Find the univariate (or multivariate if you wish - see the article) p-values of the two variables. What can you say about them?
- Create a summary table with your findings

In this exercise, the dependent variable is 'price', while the independent variables are 'size' and 'year'.

Good luck!

Import the relevant libraries

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sbs
sbs.set()

from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import f_regression
```

Load the data

In [3]:
```
data = pd.read_csv('real_estate_price_size_year.csv')
```

In [18]:
```
data
```

Out[18]:

	price	size	year
0	234314.144	643.09	2015
1	228581.528	656.22	2009
2	281626.336	487.29	2018
3	401255.608	1504.75	2015
4	458674.256	1275.46	2009
...	...	...	...
95	252460.400	549.80	2009
96	310522.592	1037.44	2009
97	383635.568	1504.75	2006
98	225145.248	648.29	2015
99	274922.856	705.29	2006

100 rows × 3 columns

In [19]:
```
data.describe()
```

Out[19]:

	price	size	year
count	100.000000	100.000000	100.000000
mean	292289.470160	853.024200	2012.600000
std	77051.727525	297.941951	4.729021
min	154282.128000	479.750000	2006.000000
25%	234280.148000	643.330000	2009.000000
50%	280590.716000	696.405000	2015.000000
75%	335723.696000	1029.322500	2018.000000
max	500681.128000	1842.510000	2018.000000

Create the regression

Declare the dependent and the independent variables

In [5]:
```
x = data[['size','year']]
y = data['price']
```

Regression

In [8]:
```
reg = LinearRegression()
reg.fit(x,y)
```

Out[8]:
```
LinearRegression()
```

Find the intercept

In [9]:
```
reg.intercept_
```

Out[9]:
```
-5772267.01746328
```

Find the coefficients

In [10]:
```
reg.coef_
```

Out[10]:
```
array([ 227.70085401, 2916.78532684])
```

Calculate the R-squared

In [11]:
```
reg.score(x,y)
```

Out[11]:
```
0.7764803683276792
```

Calculate the Adjusted R-squared

In [13]:
```
x.shape
```

Out[13]:
```
(100, 2)
```

In [16]:
```
r2 = reg.score(x,y)
n = x.shape[0]
p = x.shape[1]

adjusted_r2 = 1 - (1-r2)*(n-1)/(n-p-1)
adjusted_r2
```

Out[16]:
```
0.7718717161282499
```

Compare the R-squared and the Adjusted R-squared

```
Answer...
r2 = 0.7764803683276792
adjusted_r2 = 0.7718717161282499
```

```
It seems the the R-squared is only slightly larger than the Adjusted R-
squared, implying that we were not penalized a lot for the inclusion of 2
independent variables.
```

Compare the Adjusted R-squared with the R-squared of the simple linear regression

```
Answer...
simple linear regression - R-squared:   0.745
Multiple linear regression adjusted_r2 = 0.7718717161282499
Comparing the Adjusted R-squared with the R-squared of the simple linear
regression (when only 'size' was used - a couple of lectures ago), we realize
that 'Year' is not bringing too much value to the result
```

Making predictions

Find the predicted price of an apartment that has a size of 750 sq.ft. from 2009.

In [22]:
```
reg.predict([[750,2009]])
```

Out[22]:
```
array([258330.34465995])
```

Calculate the univariate p-values of the variables

In [23]:
```
f_regression(x,y)
```

Out[23]:
```
(array([285.92105192,   0.85525799]), array([8.12763222e-31, 3.57340758e-01]))
```

In [26]:
```
p_values = f_regression(x,y)[1]
p_values
```

Out[26]:
```
array([8.12763222e-31, 3.57340758e-01])
```

In [27]:
```
p_values.round(3)
```

Out[27]:
```
array([0.   , 0.357])
```

Create a summary table with your findings

In [31]:
```
reg_summary = pd.DataFrame(data=x.columns.values, columns=['features'])
reg_summary
```

Out[31]:

	features
0	size
1	year

In [34]:
```
reg_summary['coefficients'] = reg.coef_
reg_summary['p-values'] = p_values.round(3)
```

In [35]:
```
reg_summary
```

Out[35]:

	features	coefficients	p-values
0	size	227.700854	0.000
1	year	2916.785327	0.357

```
Answer...
It seems that 'Year' is not event significant, therefore we should remove it
from the model.
```