Feature scaling with sklearn - Exercise

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. This exercise is very similar to a previous one. This time, however, **please standardize the 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 [11]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
```

Load the data

In [12]:
```
data = pd.read_csv('real_estate_price_size_year.csv')
```

In [13]:
```
data.head()
```

Out[13]:

	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

In [14]:
```
data.describe()
```

Out[14]:

	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 [15]:
```
x = data[['size','year']]
y = data['price']
```

Scale the inputs

In [19]:
```
scaler = StandardScaler()
scaler.fit(x)
```

Out[19]: StandardScaler()

In [22]:
```
x_scaled = scaler.transform(x)
```

In [23]:
```
x_scaled
```

Out[23]:
```
array([[-0.70816415,  0.51006137],
       [-0.66387316, -0.76509206],
       [-1.23371919,  1.14763808],
       [ 2.19844528,  0.51006137],
       [ 1.42498884, -0.76509206],
       [-0.937209  , -1.40266877],
       [-0.95171405,  0.51006137],
       [-0.78328682, -1.40266877],
       [-0.57603328,  1.14763808],
       [-0.53467702, -0.76509206],
       [ 0.69939906, -0.76509206],
       [ 3.33780001, -0.76509206],
       [-0.53467702,  0.51006137],
       [ 0.52699137,  1.14763808],
       [ 1.51100715, -1.40266877],
       [ 1.77668568, -1.40266877],
       [-0.54810263,  1.14763808],
       [-0.77276222, -1.40266877],
       [-0.58004747, -1.40266877],
       [ 0.58943055,  1.14763808],
       [-0.78365788,  0.51006137],
       [-1.02322731,  0.51006137],
       [ 1.19557293,  0.51006137],
       [-1.12884431,  0.51006137],
       [-1.10378093, -0.76509206],
       [ 0.84424715,  1.14763808],
       [-0.95171405,  1.14763808],
       [ 1.62279723,  0.51006137],
       [-0.58004747,  0.51006137],
       [ 2.17014356,  0.51006137],
       [ 0.5306345 , -1.40266877],
       [-0.58004747, -1.40266877],
       [-0.8606021 , -0.76509206],
       [-1.10378093,  0.51006137],
       [ 0.015233  ,  1.14763808],
       [-0.77603429,  1.14763808],
       [-0.10057126, -1.40266877],
       [-0.95387294, -1.40266877],
       [-0.56517136,  1.14763808],
       [-0.5219598 ,  0.51006137],
       [ 0.56983186,  0.51006137],
       [-0.57603328,  1.14763808],
       [-0.10057126, -0.76509206],
       [ 1.62279723, -0.76509206],
       [ 0.69939906,  1.14763808],
       [-0.5219598 ,  0.51006137],
       [-0.7415595 , -0.76509206],
       [-0.5219598 , -1.40266877],
       [-0.7415595 ,  0.51006137],
       [-0.79600403,  0.51006137],
       [-0.69328805,  0.51006137],
       [ 0.56983186,  0.51006137],
       [ 0.56983186, -0.76509206],
       [-0.42214483, -1.40266877],
```

```
       [-0.69328805,  0.51006137],
       [ 2.21224194,  1.14763808],
       [ 0.6039356 ,  1.14763808],
       [ 1.45329055,  0.51006137],
       [-0.08495304, -1.40266877],
       [-0.95751607,  0.51006137],
       [-0.08387359,  1.14763808],
       [-0.52125142,  0.51006137],
       [ 1.18939985,  0.51006137],
       [ 0.56983186, -0.76509206],
       [-0.56517136,  0.51006137],
       [-0.08748299,  0.51006137],
       [ 0.52699137, -1.40266877],
       [-1.02285625,  0.51006137],
       [-0.56517136,  1.14763808],
       [ 2.17014356, -1.40266877],
       [ 0.56983186, -0.76509206],
       [-0.70708471, -1.40266877],
       [-0.66387316,  0.51006137],
       [-1.02285625,  0.51006137],
       [-0.56517136,  1.14763808],
       [-0.56517136,  1.14763808],
       [ 1.11464825,  1.14763808],
       [ 1.62279723, -1.40266877],
       [-0.57603328, -0.76509206],
       [ 1.13205431, -1.40266877],
       [-0.58004747, -1.40266877],
       [ 0.90847416,  0.51006137],
       [-0.58004747, -1.40266877],
       [-0.70816415,  1.14763808],
       [-0.56517136,  1.14763808],
       [ 0.52699137,  1.14763808],
       [-1.25915362, -1.40266877],
       [ 0.59162317, -0.76509206],
       [-0.84791862,  1.14763808],
       [ 1.29501685,  1.14763808],
       [-0.53467702,  1.14763808],
       [ 0.73714592,  0.51006137],
       [-0.53467702,  0.51006137],
       [-0.5219598 ,  1.14763808],
       [-0.5219598 , -1.40266877],
       [-1.02285625, -0.76509206],
       [ 0.62208377, -0.76509206],
       [ 2.19844528, -1.40266877],
       [-0.69062317,  0.51006137],
       [-0.49834693, -1.40266877]])
```

Regression

In [24]:
```
reg = LinearRegression()
reg.fit(x_scaled,y)
```

Out[24]:
```
LinearRegression()
```

Find the intercept

In [25]:
```
reg.intercept_
```

Out[25]:
```
292289.4701599997
```

Find the coefficients

In [26]:
```
reg.coef_
```

Out[26]:
```
array([67501.57614152, 13724.39708231])
```

Calculate the R-squared

In [32]:
```
reg.score(x_scaled,y)
```

Out[32]:
```
0.7764803683276793
```

Calculate the Adjusted R-squared

In [33]:
```
x.shape
```

Out[33]:
```
(100, 2)
```

In [35]:
```
r2 = reg.score(x_scaled,y)
n  = x.shape[0]
p = x.shape[1]

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

Out[35]:
```
0.77187171612825
```

Compare the R-squared and the Adjusted R-squared

```
Answer...
r2 = 0.7764803683276793
adjusted_r2= 0.77187171612825

The difference is not really significant, that's mean that we are being
penalized for the extra variable
```

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

```
Answer...
Multiple Linear Regression adjusted_r2= 0.77187171612825
Simple Linear Regression adjusted_r2 = 0.745

It means the year doesn't apport so much to the analysis
```

Making predictions

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

In [54]:
```
new_data = [[750,2009]]
new_data_scaled = scaler.transform(new_data)
```

In [56]:
```
reg.predict(new_data_scaled)
```

Out[56]:
```
array([258330.34465995])
```

Calculate the univariate p-values of the variables

In [57]:
```
from sklearn.feature_selection import f_regression
```

In [58]:
```
f_regression(x,y)
```

Out[58]:
```
(array([285.92105192,   0.85525799]), array([8.12763222e-31, 3.57340758e-01]))
```

In [59]:
```
p_values = f_regression(x,y)[1]
p_values
```

Out[59]:
```
array([8.12763222e-31, 3.57340758e-01])
```

In [60]:
```
p_values.round(3)
```

Out[60]:
```
array([0.   , 0.357])
```

Create a summary table with your findings

In [63]:
```
reg_summary = pd.DataFrame(data=x.columns.values, columns=['features'])
reg_summary
```

Out[63]:

	features
0	size
1	year

In [64]:
```python
reg_summary['Coefficients'] = reg.coef_
reg_summary['p_values'] = p_values.round(3)
reg_summary
```

Out[64]:

	features	Coefficients	p_values
0	size	67501.576142	0.000
1	year	13724.397082	0.357

```
Answer...
we have been penalized for the variable year
Note that this dataset is extremely clean and probably artificially created,
therefore standardization does not really bring any value to it.
```