Feature scaling with sklearn

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

- create a multiple linear regression
- standardize the data
- 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. Describe
- Create a summary table with your findings

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

Import the relevant libraries

In [1]:
```
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
```

Load the data

In [2]:
```
data = pd.read_csv('real_estate_price_size_year.csv')
data.head()
```

Out[2]:

	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 [3]:
```
data.describe()
```

Out[3]:

	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 [4]:
```
x = data[['size','year']]
y = data['price']
```

Scale the inputs

In [5]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(x)
x_scaled = scaler.transform(x)
```

Regression

In [6]:
```
reg = LinearRegression()
reg.fit(x_scaled,y)
```

Out[6]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Find the intercept

In [7]:
```
reg.intercept_
```

Out[7]:
```
292289.4701599997
```

Find the coefficients

In [8]:
```
reg.coef_
```

Out[8]:
```
array([67501.57614152, 13724.39708231])
```

Calculate the R-squared

In [9]:
```
reg.score(x_scaled,y)
```

Out[9]:
```
0.7764803683276793
```

Calculate the Adjusted R-squared

In [10]:
```
# Let's use the handy function we created
def adj_r2(x,y):
    r2 = reg.score(x,y)
    n = x.shape[0]
    p = x.shape[1]
    adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)
    return adjusted_r2
```

In [11]:
```
adj_r2(x_scaled,y)
```

Out[11]:
```
0.7718717161282 5
```

Compare the R-squared and the Adjusted R-squared

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

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 [12]:
```
new_data = [[750,2009]]
new_data_scaled = scaler.transform(new_data)
```

In [13]:
```
reg.predict(new_data_scaled)
```

Out[13]:
```
array([258330.34465995])
```

Calculate the univariate p-values of the variables

In [14]:
```
from sklearn.feature_selection import f_regression
```

In [15]:
```
f_regression(x_scaled,y)
```

Out[15]:
```
(array([285.92105192,   0.85525799]), array([8.12763222e-31, 3.57340758e-01]))
```

In [16]:
```
p_values = f_regression(x,y)[1]
p_values
```

Out[16]:
```
array([8.12763222e-31, 3.57340758e-01])
```

In [17]:
```
p_values.round(3)
```

Out[17]:
```
array([0.   , 0.357])
```

Create a summary table with your findings

In [18]:
```
reg_summary = pd.DataFrame(data = x.columns.values, columns=['Features'])
reg_summary ['Coefficients'] = reg.coef_
reg_summary ['p-values'] = p_values.round(3)
reg_summary
```

Out[18]:

	Features	Coefficients	p-values
0	size	67501.576142	0.000
1	year	13724.397082	0.357

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

This dataset is extremely clean and probably artificially created, therefore standardization does not really bring any value to it.