Adjusted R-squared - Exercise

Using the code from the lecture, create a function which will calculate the adjusted R-squared for you, given the independent variable(s) (x) and the dependent variable (y).

Check if you function is working properly.

Please solve the exercise at the bottom of the notebook (in order to check if it is working you must run all previous cells).

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('1.02. Multiple linear regression.csv')
data.head()
```

Out[2]:

	SAT	GPA	Rand 1,2,3
0	1714	2.40	1
1	1664	2.52	3
2	1760	2.54	3
3	1685	2.74	3
4	1693	2.83	2

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

Out[3]:

	SAT	GPA	Rand 1,2,3
count	84.000000	84.000000	84.000000
mean	1845.273810	3.330238	2.059524
std	104.530661	0.271617	0.855192
min	1634.000000	2.400000	1.000000
25%	1772.000000	3.190000	1.000000
50%	1846.000000	3.380000	2.000000
75%	1934.000000	3.502500	3.000000
max	2050.000000	3.810000	3.000000

Create the multiple linear regression

Declare the dependent and independent variables

In [4]:
```
x = data[['SAT','Rand 1,2,3']]
y = data['GPA']
```

Regression itself

In [5]:
```
reg = LinearRegression()
reg.fit(x,y)
```

Out[5]:
```
LinearRegression()
```

In [6]:
```
reg.coef_
```

Out[6]:
```
array([ 0.00165354, -0.00826982])
```

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

Out[7]:
```
0.29603261264909486
```

Calculating the R-squared

In [8]:
```
reg.score(x,y)
```

Out[8]:
```
0.40668119528142843
```

Formula for Adjusted R^2

$$R^2_{adj.} = 1 - (1 - R^2) * \frac{n-1}{n-p-1}$$

In [9]:
```
x.shape
```

Out[9]:
```
(84, 2)
```

In [10]:
```
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[10]:
```
0.39203134825134023
```

Adjusted R^2 function

In [11]:
```
def adjusted_r2_funtion(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 [12]:
```
adjusted_r2_funtion(x,y)
```

Out[12]:
```
0.39203134825134023
```

In [13]:
```
from sklearn.feature_selection import f_regression
```

In [14]:
```
f_regression(x,y)
```

Out[14]:
```
(array([56.04804786,  0.17558437]), array([7.19951844e-11, 6.76291372e-01]))
```

In [15]:
```
# F-statistics array([56.04804786,  0.17558437]
# p-values array([7.19951844e-11, 6.76291372e-01]
```

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

Out[16]:
```
array([7.19951844e-11, 6.76291372e-01])
```

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

Out[17]:
```
array([0.   , 0.676])
```

In [20]:
```
reg_summary = pd.DataFrame(data=['SAT','Rand 1,2,3'], columns=['Features'])
reg_summary
```

Out[20]:

	Features
0	SAT
1	Rand 1,2,3

In [21]:
```
reg_summary = pd.DataFrame(data= x.columns.values, columns=['Features'])#the same
reg_summary
```

Out[21]:

	Features
0	SAT
1	Rand 1,2,3

In [22]:
```
reg_summary['Coefficients'] = reg.coef_
reg_summary['p-values'] = p_values.round(3)
```

In [23]:
```
reg_summary
```

Out[23]:

	Features	Coefficients	p-values
0	SAT	0.001654	0.000
1	Rand 1,2,3	-0.008270	0.676

In [ ]:
```

```