In [2]:

```
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.feature_selection import f_regression
```

In [4]:

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

Out[4]:

	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 [5]:

```
data.describe()
```

Out[5]:

	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

In [6]:

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

Standarization

In [9]:

```
from sklearn.preprocessing import StandardScaler
```

In [11]:

```
scaler = StandardScaler()
```

In [12]:
```
scaler.fit(x)
```

Out[12]:
```
StandardScaler()
```

In [13]:
```
x_scale = scaler.transform(x)
```

In [14]:
```
x_scale
```

Out[14]:
```
array([[-1.26338288, -1.24637147],
       [-1.74458431,  1.10632974],
       [-0.82067757,  1.10632974],
       [-1.54247971,  1.10632974],
       [-1.46548748, -0.07002087],
       [-1.68684014, -1.24637147],
       [-0.78218146, -0.07002087],
       [-0.78218146, -1.24637147],
       [-0.51270866, -0.07002087],
       [ 0.04548499,  1.10632974],
       [-1.06127829,  1.10632974],
       [-0.67631715, -0.07002087],
       [-1.06127829, -1.24637147],
       [-1.28263094,  1.10632974],
       [-0.6955652 , -0.07002087],
       [ 0.25721362, -0.07002087],
       [-0.86879772,  1.10632974],
       [-1.64834403, -0.07002087],
       [-0.03150724,  1.10632974],
       [-0.57045283,  1.10632974],
       [-0.81105355,  1.10632974],
       [-1.18639066,  1.10632974],
       [-1.75420834,  1.10632974],
       [-1.52323165, -1.24637147],
       [ 1.23886453, -1.24637147],
       [-0.18549169, -1.24637147],
       [-0.5608288 , -1.24637147],
       [-0.23361183,  1.10632974],
       [ 1.68156984, -1.24637147],
       [-0.4934606 , -0.07002087],
       [-0.73406132, -1.24637147],
       [ 0.85390339, -1.24637147],
       [-0.67631715, -1.24637147],
       [ 0.09360513,  1.10632974],
       [ 0.33420585, -0.07002087],
       [ 0.03586096, -0.07002087],
       [-0.35872421,  1.10632974],
       [ 1.04638396,  1.10632974],
       [-0.65706909,  1.10632974],
       [-0.13737155, -0.07002087],
       [ 0.18984542,  1.10632974],
       [ 0.04548499, -1.24637147],
       [ 1.1618723 ,  1.10632974],
       [-1.37887123, -1.24637147],
       [ 1.39284898, -1.24637147],
       [ 0.76728713, -0.07002087],
       [-0.20473975, -0.07002087],
       [ 1.06563201, -1.24637147],
       [ 0.11285319, -1.24637147],
       [ 1.28698467,  1.10632974],
       [-0.41646838,  1.10632974],
       [ 0.09360513, -1.24637147],
       [ 0.59405462, -0.07002087],
       [-2.03330517, -0.07002087],
       [ 0.32458182, -1.24637147],
       [ 0.40157405, -1.24637147],
       [-1.10939843, -0.07002087],
       [ 1.03675993, -1.24637147],
       [-0.61857297, -0.07002087],
       [ 0.44007016, -0.07002087],
       [ 1.14262424, -1.24637147],
       [-0.35872421,  1.10632974],
       [ 0.45931822,  1.10632974],
       [ 1.88367444,  1.10632974],
       [ 0.45931822, -1.24637147],
       [-0.12774752, -0.07002087],
       [ 0.04548499,  1.10632974],
       [ 0.85390339, -0.07002087],
       [ 0.15134931, -0.07002087],
       [ 0.8250313 ,  1.10632974],
       [ 0.84427936,  1.10632974],
       [-0.64744506, -1.24637147],
       [ 1.24848856, -1.24637147],
       [ 0.85390339,  1.10632974],
       [ 1.69119387,  1.10632974],
       [ 1.6334497 ,  1.10632974],
       [ 1.46021718, -1.24637147],
       [ 1.68156984, -0.07002087],
       [-0.02188321,  1.10632974],
       [ 0.87315144,  1.10632974],
       [-0.33947615, -1.24637147],
       [ 1.3639769 ,  1.10632974],
       [ 1.12337618, -1.24637147],
       [ 1.97029069, -0.07002087]])
```

In [15]:

```
reg = LinearRegression()
reg.fit(x_scale,y)
```

Out[15]:

```
LinearRegression()
```

In [16]:

```
reg.coef_
```

Out[16]:

```
array([ 0.17181389, -0.00703007])
```

In [18]:

```
reg.intercept_
```

Out[18]:

```
3.330238095238095
```

In [19]:

```
reg_summary = pd.DataFrame([['Intercept'],['SAT'],['Rand 1,2,3']], columns=['Feat
reg_summary ['Weight'] = reg.intercept_, reg.coef_[0], reg.coef_[1]
```

In [20]:
```
reg_summary
```

Out[20]:

	Features	Weight
0	Intercept	3.330238
1	SAT	0.171814
2	Rand 1,2,3	-0.007030

In [21]:
```
#biggr the number, bigger the impact
#weight is known as coeficients
# intercept is known as bias - coenfficient with standarization
```

In [22]:
```
#same as above
reg_summary = pd.DataFrame([['Bias'],['SAT'],['Rand 1,2,3']], columns=['Features'
reg_summary ['Weight'] = reg.intercept_, reg.coef_[0], reg.coef_[1]
reg_summary
```

Out[22]:

	Features	Weight
0	Bias	3.330238
1	SAT	0.171814
2	Rand 1,2,3	-0.007030

In [23]:
```
new_data = pd.DataFrame(data=[[1700,2],[1800,1]], columns=['SAT','Rand 1,2,3'])
new_data
```

Out[23]:

	SAT	Rand 1,2,3
0	1700	2
1	1800	1

In [24]:
```
reg.predict(new_data)
```

Out[24]:
```
array([295.39979563, 312.58821497])
```

In [26]:
```
## the result above, doesn't make sense at all, it is because we need to standari
```

In [28]:
```
new_data_scaled = scaler.transform(new_data)
new_data_scaled
```

Out[28]:
```
array([[-1.39811928, -0.07002087],
       [-0.43571643, -1.24637147]])
```

```
In [29]: reg.predict(new_data_scaled)

Out[29]: array([3.09051403, 3.26413803])

In [ ]: ## Now the results without the random var

In [32]: reg_simple = LinearRegression()
         x_simple_matrix = x_scale[:,0].reshape(-1,1)
         reg_simple.fit(x_simple_matrix,y)

Out[32]: LinearRegression()

In [35]: reg_simple.predict(new_data_scaled[:,0].reshape(-1,1))

Out[35]: array([3.08970998, 3.25527879])

In [ ]: ##This show us that the random var is not relevant

In [ ]:

In [ ]:
```