Simple linear regression

Import the relevant libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
# We can override the default matplotlib styles with those of Seaborn
#import seaborn as sns
#sns.set()
```

Load the data

In [2]:
```
# Load the data from a .csv in the same folder
data = pd.read_csv('1.01.Simple_linear_regression.csv')
```

In [3]:
```
# Let's check what's inside this data frame
data
```

Out[3]:

	SAT	GPA
0	1714	2.40
1	1664	2.52
2	1760	2.54
3	1685	2.74
4	1693	2.83
...	...	...
79	1936	3.71
80	1810	3.71
81	1987	3.73
82	1962	3.76
83	2050	3.81

84 rows × 2 columns

In [4]:
```
# This method gives us very nice descriptive statistics. We don't need this as of now, but will later on!
data.describe()
```

Out[4]:

	SAT	GPA
count	84.000000	84.000000
mean	1845.273810	3.330238
std	104.530661	0.271617
min	1634.000000	2.400000
25%	1772.000000	3.190000
50%	1846.000000	3.380000
75%	1934.000000	3.502500
max	2050.000000	3.810000

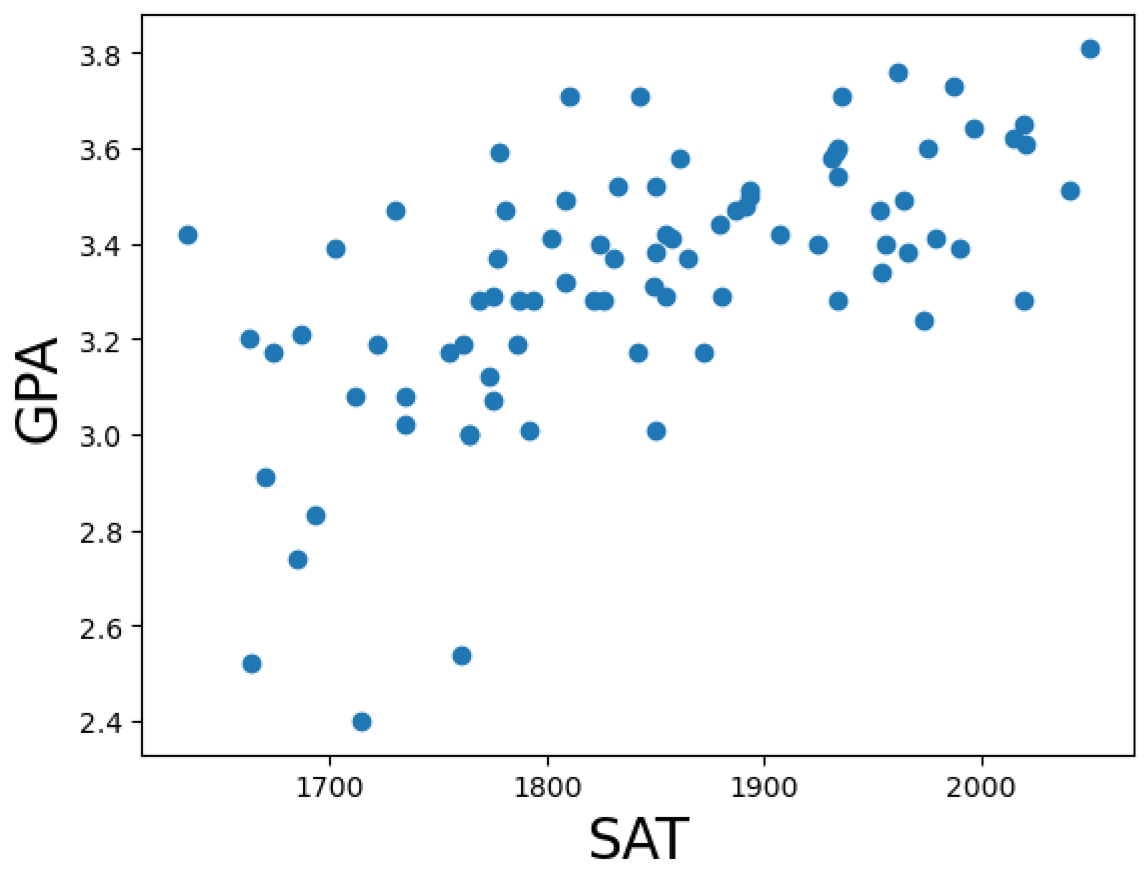
Create regression

Define the dependent and the independent variables

In [5]:
```
# Following the regression equation, our dependent variable (y) is the GPA
y = data ['GPA']
# Similarly, our independent variable (x) is the SAT score
x1 = data ['SAT']
```

Explore the data

In [6]:
```
# Plot a scatter plot (first we put the horizontal axis, then the vertical axis)
plt.scatter(x1,y)
# Name the axes
plt.xlabel('SAT', fontsize = 20)
plt.ylabel('GPA', fontsize = 20)
# Show the plot
plt.show()
```



Regression itself

In [7]:
```
# Add a constant. Essentially, we are adding a new column (equal in lenght to x), which consists only of 1s
x = sm.add_constant(x1)
# Fit the model, according to the OLS (ordinary least squares) method with a dependent variable y and an idepen
results = sm.OLS(y,x).fit()
# Print a nice summary of the regression. That's one of the strong points of statsmodels -> the summaries
results.summary()
```

Out[7]:

OLS Regression Results

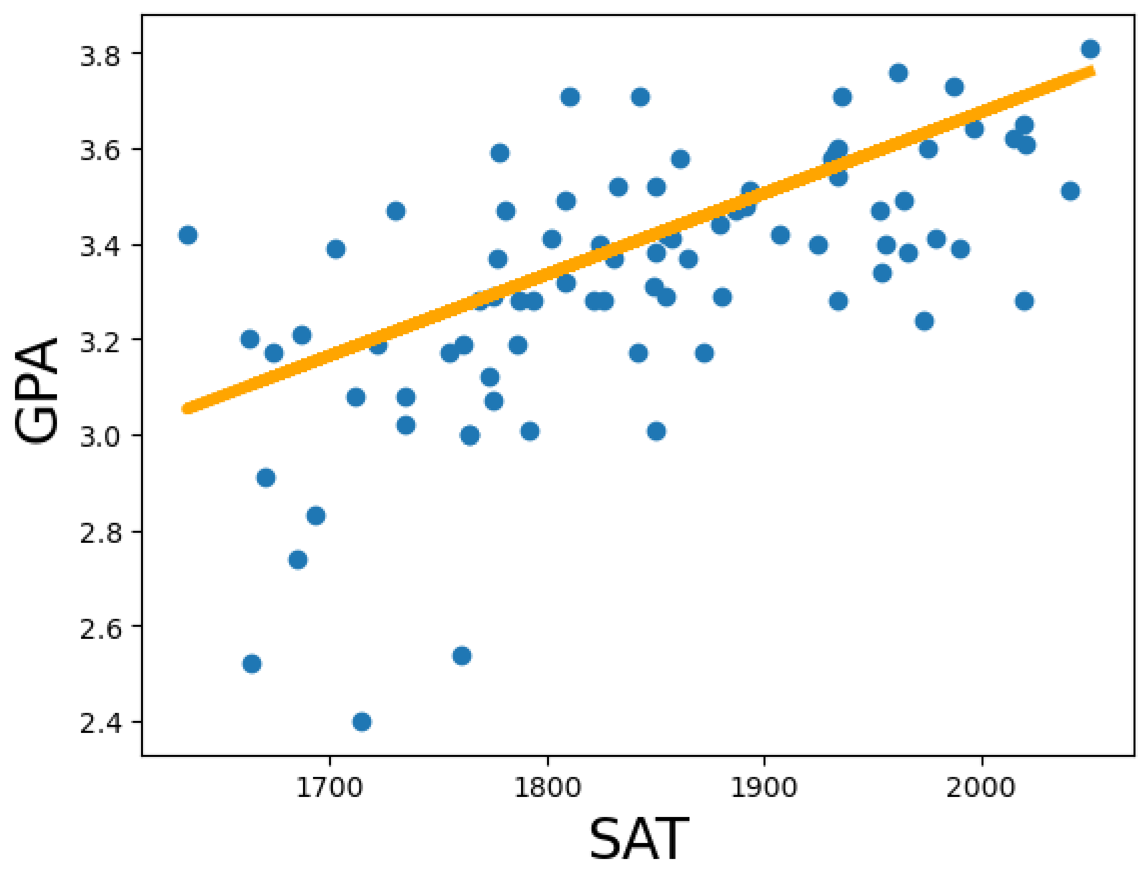
Dep. Variable:	GPA	**R-squared:**	0.406
Model:	OLS	**Adj. R-squared:**	0.399
Method:	Least Squares	**F-statistic:**	56.05
Date:	Wed, 21 Sep 2022	**Prob (F-statistic):**	7.20e-11
Time:	14:40:06	**Log-Likelihood:**	12.672
No. Observations:	84	**AIC:**	-21.34
Df Residuals:	82	**BIC:**	-16.48
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.2750	0.409	0.673	0.503	-0.538	1.088
SAT	0.0017	0.000	7.487	0.000	0.001	0.002

Omnibus:	12.839	**Durbin-Watson:**	0.950
Prob(Omnibus):	0.002	**Jarque-Bera (JB):**	16.155
Skew:	-0.722	**Prob(JB):**	0.000310
Kurtosis:	4.590	**Cond. No.**	3.29e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.29e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [8]:
```
# Create a scatter plot
plt.scatter(x1,y)
# Define the regression equation, so we can plot it later
yhat = 0.0017*x1 + 0.275
# Plot the regression line against the independent variable (SAT)
fig = plt.plot(x1,yhat, lw=4, c='orange', label ='regression line')
# Label the axes
plt.xlabel('SAT', fontsize = 20)
plt.ylabel('GPA', fontsize = 20)
plt.show()
```



In [ ]: