In [37]:
```
import numpy as np # numpy is a library that performs mathematical operations
import pandas as pd # pandas is used for data manipulation and analysis
import matplotlib.pyplot as plt # matplotlib (a component of numpy) is what plots the graphs
import statsmodels.api as sm # statsmodels explores data, estimates statistic models, and performs statistical 
```

In [38]:
```
data = pd.read_csv('data.csv') # defining what the command "data" means - to read the csv file
```

In [39]:
```
data # we are shown what the raw data given to us is, which we will use to train and test the model
```

Out[39]:

	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 [40]:
```
data.describe() # gives us descriptive statistics of the data
```

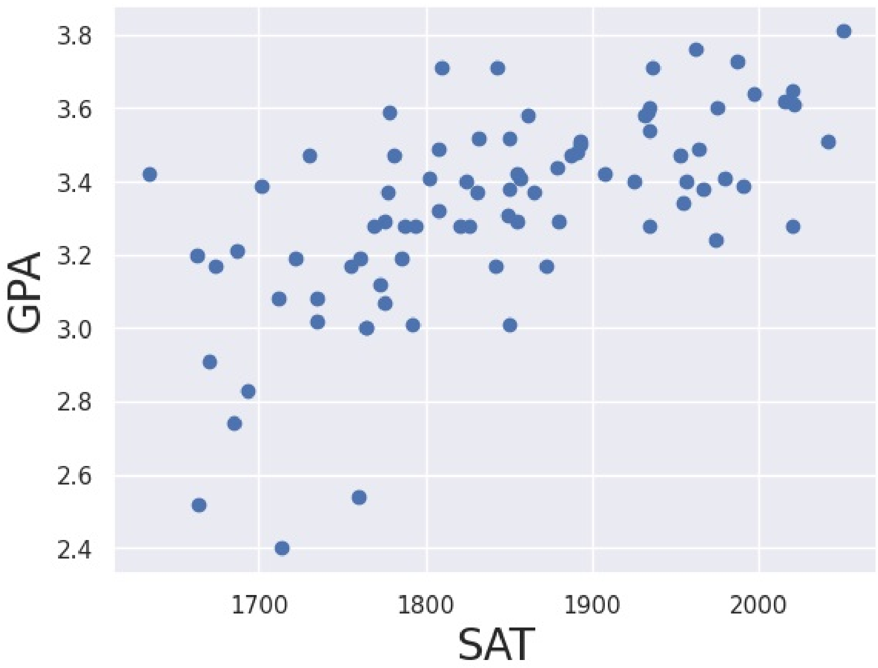
Out[40]:

	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

In [41]:
```
y = data ['GPA'] # GPA is the dependent variable so is on the y-axis of the graph
x1 = data ['SAT'] # SAT is the independent variable so is on the x-axis
```

In [42]:
```
plt.scatter(x1,y) # plots a scatter plot
plt.xlabel('SAT', fontsize = 20) # the x-axis is labeled SAT
plt.ylabel('GPA', fontsize = 20) # the y-axis is labeled GPA
plt.show # shows the plot
```

Out[42]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [43]:
```
x = sm.add_constant(x1)
results = sm.OLS(y,x).fit()
results.summary()
```

Out[43]:

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:	Sun, 03 Mar 2024	**Prob (F-statistic):**	7.20e-11
Time:	11:34:45	**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 [44]:

```
x = sm.add_constant(x1)
results = sm.OLS(y,x).fit()
results.summary()
```

Out[44]:

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:	Sun, 03 Mar 2024	**Prob (F-statistic):**	7.20e-11
Time:	11:34:45	**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

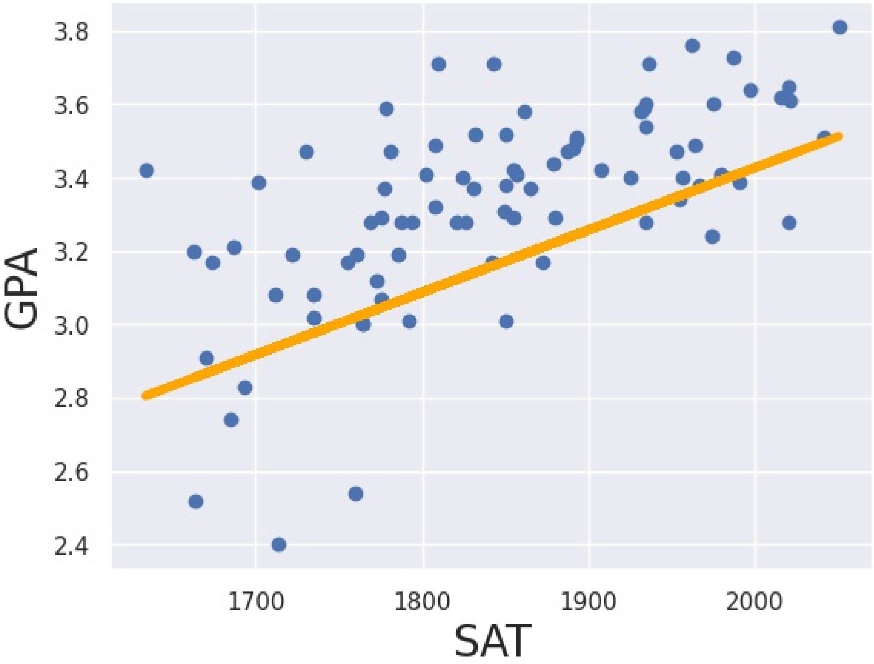
Omnibus:	12.839	**Durbin-Watson:**	0.950
Prob(Omnibus):	0.002	**Jarque-Bera (JB):**	16.155
Skew:	-0.722	**Prob(JB):**	0.000310
Kurtosis:	4.590	**Cond. No.**	3.29e+04

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 [45]:

```
plt.scatter(x1,y) # plot a scatter plot
yhat = 0.0017*x1 + 0.0275
#  define the regression equation, to plot it later. The 0.0017 and 0.0275 come from the "coef" column from the
```

```
fig = plt.plot(x1,yhat, lw=4, c='orange', label = 'regression line') # plot the regression line against the ind
plt.xlabel('SAT', fontsize = 20) # label the axes
plt.ylabel('GPA', fontsize = 20)
plt.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js