In [29]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

In [30]:
```
raw_data = pd.read_csv(r"C:\Users\krish\OneDrive\Desktop\python\dummies_data.csv")
raw_data
```

Out[30]:

	SAT	GPA	Attendance
0	1714	2.40	No
1	1664	2.52	No
2	1760	2.54	No
3	1685	2.74	No
4	1693	2.83	No
...	...	...	...
79	1936	3.71	Yes
80	1810	3.71	Yes
81	1987	3.73	No
82	1962	3.76	Yes
83	2050	3.81	Yes

84 rows × 3 columns

In [31]:
```
data = raw_data.copy()
```

In [32]:
```
data['Attendance'] = data['Attendance'].map({'Yes':1,'No': 0})
data
```

Out[32]:

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

84 rows × 3 columns

In [33]:
```
data.describe()
```

Out[33]:

	SAT	GPA	Attendance
count	84.000000	84.000000	84.000000
mean	1845.273810	3.330238	0.464286
std	104.530661	0.271617	0.501718
min	1634.000000	2.400000	0.000000
25%	1772.000000	3.190000	0.000000
50%	1846.000000	3.380000	0.000000
75%	1934.000000	3.502500	1.000000
max	2050.000000	3.810000	1.000000

Pridcition Base on One Independent Variable

Prediction Base on One Independent Variable

In [34]:
```
y = data['GPA']
x1 = data['SAT']
```

In [35]:
```
x = sm.add_constant(x1)
result = sm.OLS(y,x).fit()
result.summary()
```

Out[35]:

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:	Tue, 26 Dec 2023	**Prob (F-statistic):**	7.20e-11
Time:	14:24:26	**Log-Likelihood:**	12.672
No. Observations:	84	**AIC:**	-21.34
Df Residuals:	82	**BIC:**	-16.48
Df Model:	1		
Covariance Type:	nonrobust		

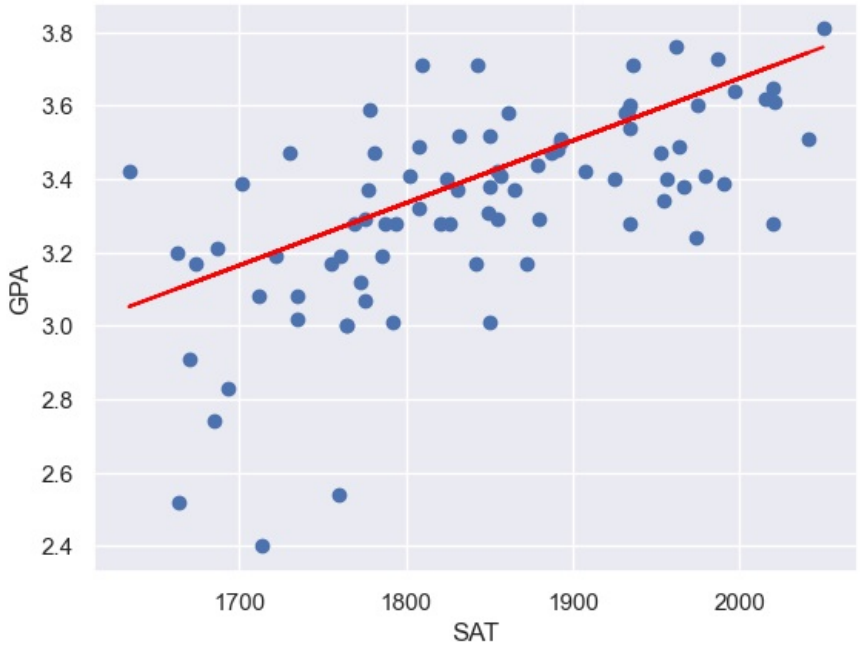
	coef	std err	t	P>|t|	[0.025	0.975]
const	0.2750	0.409	0.673	0.503	-0.538	1.088
SAT	0.0017	0.000	7.487	0.000	0.001	0.002

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 [36]:
```
plt.scatter(x1,y)
y_hat = 0.2750 + 0.0017*x1

fig = plt.plot(x1, y_hat , c = 'Red')
plt.xlabel('SAT')
plt.ylabel('GPA')
plt.show()
```



Pridiction Base on Two Indepedent Variable

Prediction Base on Two Independent Variable

In [37]:
```
y = data['GPA']
x1 = data[['SAT','Attendance']]
```

In [38]:
```
x = murari.add_constant(x1)
results = murari.OLS(y,x).fit()
results.summary()
```

Out[38]:

OLS Regression Results

Dep. Variable:	GPA	**R-squared:**	0.565
Model:	OLS	**Adj. R-squared:**	0.555
Method:	Least Squares	**F-statistic:**	52.70
Date:	Tue, 26 Dec 2023	**Prob (F-statistic):**	2.19e-15
Time:	14:24:34	**Log-Likelihood:**	25.798
No. Observations:	84	**AIC:**	-45.60
Df Residuals:	81	**BIC:**	-38.30
Df Model:	2		
Covariance Type:	nonrobust		

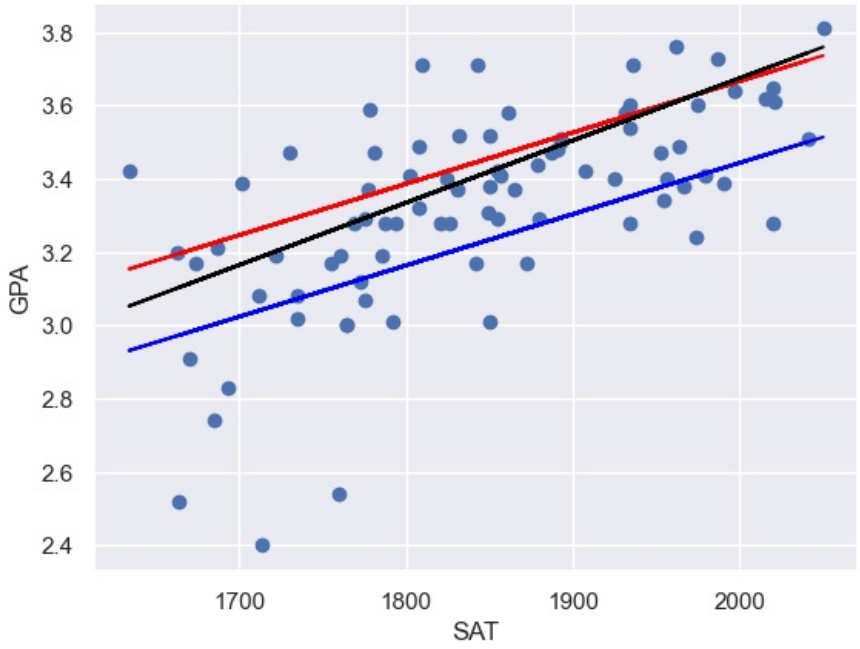
	coef	std err	t	P>|t|	[0.025	0.975]
const	0.6439	0.358	1.797	0.076	-0.069	1.357
SAT	0.0014	0.000	7.141	0.000	0.001	0.002
Attendance	0.2226	0.041	5.451	0.000	0.141	0.304

Omnibus:	19.560	**Durbin-Watson:**	1.009
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	27.189
Skew:	-1.028	**Prob(JB):**	1.25e-06
Kurtosis:	4.881	**Cond. No.**	3.35e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.35e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [39]:
```
plt.scatter(data['SAT'],y)
y_hatno = 0.6439 + 0.0014*data['SAT']
y_hatyes = 0.8665 + 0.0014*data['SAT']

fig = plt.plot(data['SAT'], y_hatno  , c = 'Blue')
fig = plt.plot(data['SAT'], y_hatyes , c = 'red')
fig = plt.plot(data['SAT'], y_hat  , c = 'Black')
plt.xlabel('SAT')
plt.ylabel('GPA')
plt.show()
```



Pridicted the Value base on SAT and Attendance

In [40]:
```
x
```

Out[40]:

	const	SAT	Attendance
0	1.0	1714	0
1	1.0	1664	0
2	1.0	1760	0
3	1.0	1685	0
4	1.0	1693	0
...	...	...	...
79	1.0	1936	1
80	1.0	1810	1
81	1.0	1987	0
82	1.0	1962	1
83	1.0	2050	1

84 rows × 3 columns

In [48]:
```
new_data = pd.DataFrame({'const': 1,'SAT': [1700, 1670], 'Attendance': [0, 1]})
new_data = new_data[['const','SAT','Attendance']]
new_data
```

Out[48]:

	const	SAT	Attendance
0	1	1700	0
1	1	1670	1

In [51]:
```
new_data.rename(index={0: 'A',1:'B'})
```

Out[51]:

	const	SAT	Attendance
A	1	1700	0
B	1	1670	1

In [50]:
```
predictions = results.predict(new_data)
predictions
```

Out[50]:
```
0    3.023513
1    3.204163
dtype: float64
```

In [ ]:
```

```

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