Basics of Logistic Regression

Import the Relevant Libraries

In [1]:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

Load the Data

In [2]:

```
raw_data = pd.read_csv('Admit.csv')
raw_data
```

Out[2]:

	SAT	Admitted
0	1363	No
1	1792	Yes
2	1954	Yes
3	1653	No
4	1593	No
...	...	...
163	1722	Yes
164	1750	Yes
165	1555	No
166	1524	No
167	1461	No

168 rows × 2 columns

In [3]:

```
# Replace All "No" Entries With 0 and All "Yes" Wntries With 1
data = raw_data.copy()
data['Admitted'] = data['Admitted'].map({'Yes': 1, 'No': 0})
data
```

Out[3]:

	SAT	Admitted
0	1363	0
1	1792	1
2	1954	1

	SAT	Admitted
3	1653	0
4	1593	0
...	...	...
163	1722	1
164	1750	1
165	1555	0
166	1524	0
167	1461	0

168 rows × 2 columns

Variables

In [4]:

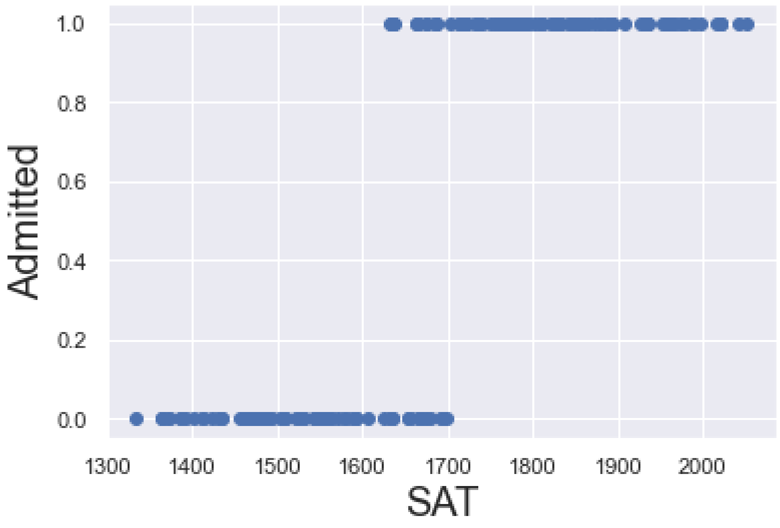
```
# Create the Dependent and Independent Variables
y = data['Admitted']
x1 = data['SAT']
```

Let's Plot the Data

Scatterplot

In [5]:

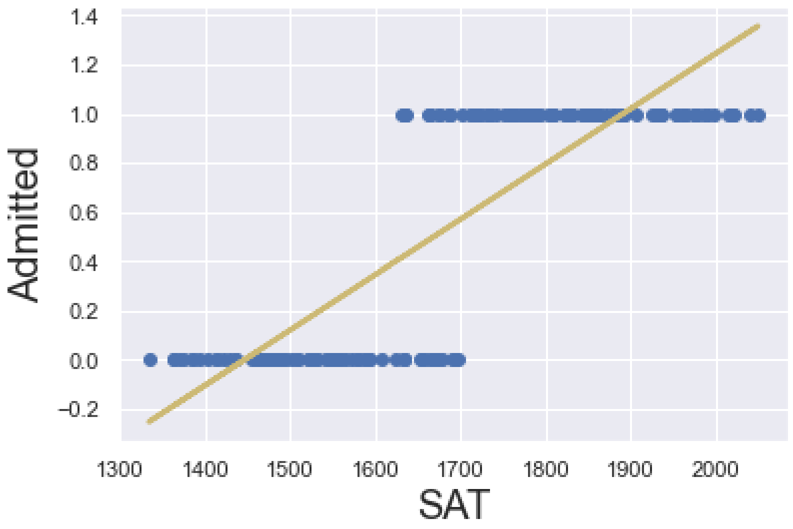
```
# Create a Scatterplot of x1 (SAT, No Constant) and y (Admitted)
plt.scatter(x1, y, color = 'C0')

# Don't Forget to Label the Axes
plt.xlabel('SAT', fontsize = 20)
plt.ylabel('Admitted', fontsize = 20)
plt.show()
```



Plot With a Regression Line

In [6]:

```
# Create a Linear Regression On the Data In Order to Estimate the Coefficients and be Al
# The Data Is Not Linear, so the Linear Regression Doesn't Make Much Sense
x = sm.add_constant(x1)

# We'll Call It reg_lin, Instead of reg, as We Will be Dealing With Logistic Regressions
reg_lin = sm.OLS(y, x)

# We'll Segment It Into Regression and Fitted Regression (Results) to Use the Results as
results_lin = reg_lin.fit()

# Create a Scatterplot
plt.scatter(x1, y, color = 'C0')

# Plot the Regression Line. The Coefficients Are Coming From results_lin.params.
y_hat = x1 * results_lin.params[1] + results_lin.params[0]
plt.plot(x1, y_hat, lw = 2.5, color = 'C8')
plt.xlabel('SAT', fontsize = 20)
plt.ylabel('Admitted', fontsize = 20)
plt.show()
```

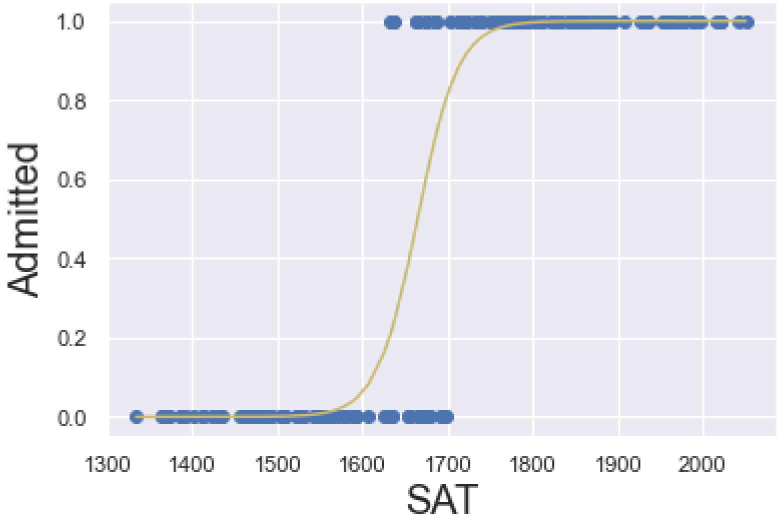


Plot a Logistic Regression Curve

In [7]:

```
reg_log = sm.Logit(y, x)
results_log = reg_log.fit()

def f(x, b0, b1):
    return np.array(np.exp(b0 + x * b1) / (1 + np.exp(b0 + x * b1)))

f_sorted = np.sort(f(x1, results_log.params[0], results_log.params[1]))
x_sorted = np.sort(np.array(x1))

plt.scatter(x1, y, color = 'C0')
plt.xlabel('SAT', fontsize = 20)
plt.ylabel('Admitted', fontsize = 20)
plt.plot(x_sorted, f_sorted, color = 'C8')
plt.show()
```

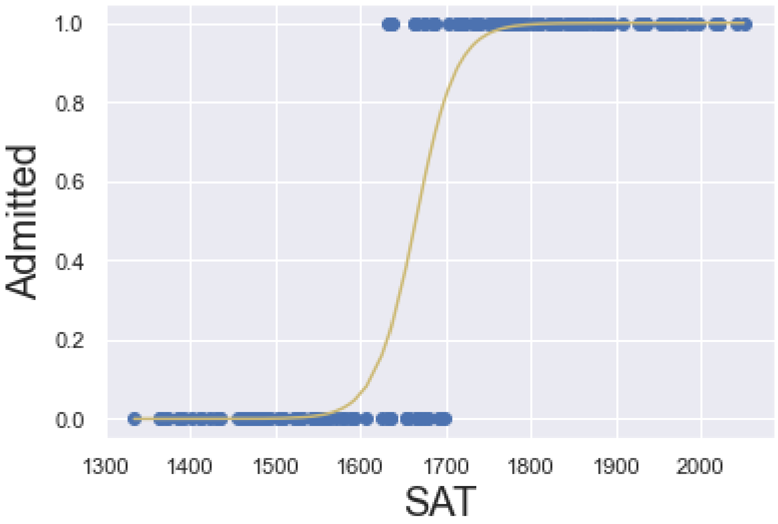
```
Optimization terminated successfully.
         Current function value: 0.137766
         Iterations 10
```



In [8]:

```python
# Creating a Logistic Regression
reg_log = sm.Logit(y, x)

# Fitting the Regression
results_log = reg_log.fit()

# Creating a Logistic Regression Function, Depending On the Input and Coefficients
def f(x, b0, b1):
    return np.array(np.exp(b0 + x * b1) / (1 + np.exp(b0 + x * b1)))

# Sorting the y and x, so We Can Plot the Curve
f_sorted = np.sort(f(x1, results_log.params[0], results_log.params[1]))
x_sorted = np.sort(np.array(x1))
plt.scatter(x1, y, color = 'C0')
plt.xlabel('SAT', fontsize = 20)
plt.ylabel('Admitted', fontsize = 20)

# Plotting the Curve
plt.plot(x_sorted, f_sorted, color = 'C8')
plt.show()
```

```
Optimization terminated successfully.
         Current function value: 0.137766
         Iterations 10
```


1.0
0.8
0.6
0.4
0.2
0.0
Admitted
1300
1400
1500
1600
1700
1800
1900
2000
SAT