Here is how a probability can be "extracted" from a logit regression model with M predictor variables ($M+1$ variables in all, namely β_0,...,β_M).

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \sum_{j=1}^{M} \beta_j v_j$$

Here, $\frac{p}{1-p}$ is the odds of the event occurring, and p is the probability of the event.

To solve for p in terms of the other variables, start by exponentiating both sides to rem

logarithm:

$$\frac{p}{1-p} = e^{\beta_0 + \sum_{j=1}^{M} \beta_j v_j}$$

This gives you the odds of the event. The next step is to solve for p:

$$\frac{p}{1-p} = e^{\beta_0 + \sum_{j=1}^{M} \beta_j v_j}$$

$$p = (1-p) \cdot e^{\beta_0 + \sum_{j=1}^{M} \beta_j v_j}$$

$$p = e^{\beta_0 + \sum_{j=1}^{M} \beta_j v_j} - p \cdot e^{\beta_0 + \sum_{j=1}^{M} \beta_j v_j}$$

$$p = e^{\beta_0+\sum_{j=1}^{M} \beta_j v_j} - p \cdot e^{\beta_0+\sum_{j=1}^{M} \beta_j v_j}$$

To isolate p, rearrange the equation:

$$p + p \cdot e^{\beta_0+\sum_{j=1}^{M} \beta_j v_j} = e^{\beta_0+\sum_{j=1}^{M} \beta_j v_j}$$

$$p(1 + e^{\beta_0+\sum_{j=1}^{M} \beta_j v_j}) = e^{\beta_0+\sum_{j=1}^{M} \beta_j v_j}$$

$$p = \frac{e^{\beta_0+\sum_{j=1}^{M} \beta_j v_j}}{1+e^{\beta_0+\sum_{j=1}^{M} \beta_j v_j}}$$

This expression simplifies to:

$$p = \frac{1}{1+e^{-\beta_0-\sum_{j=1}^{M} \beta_j v_j}}$$

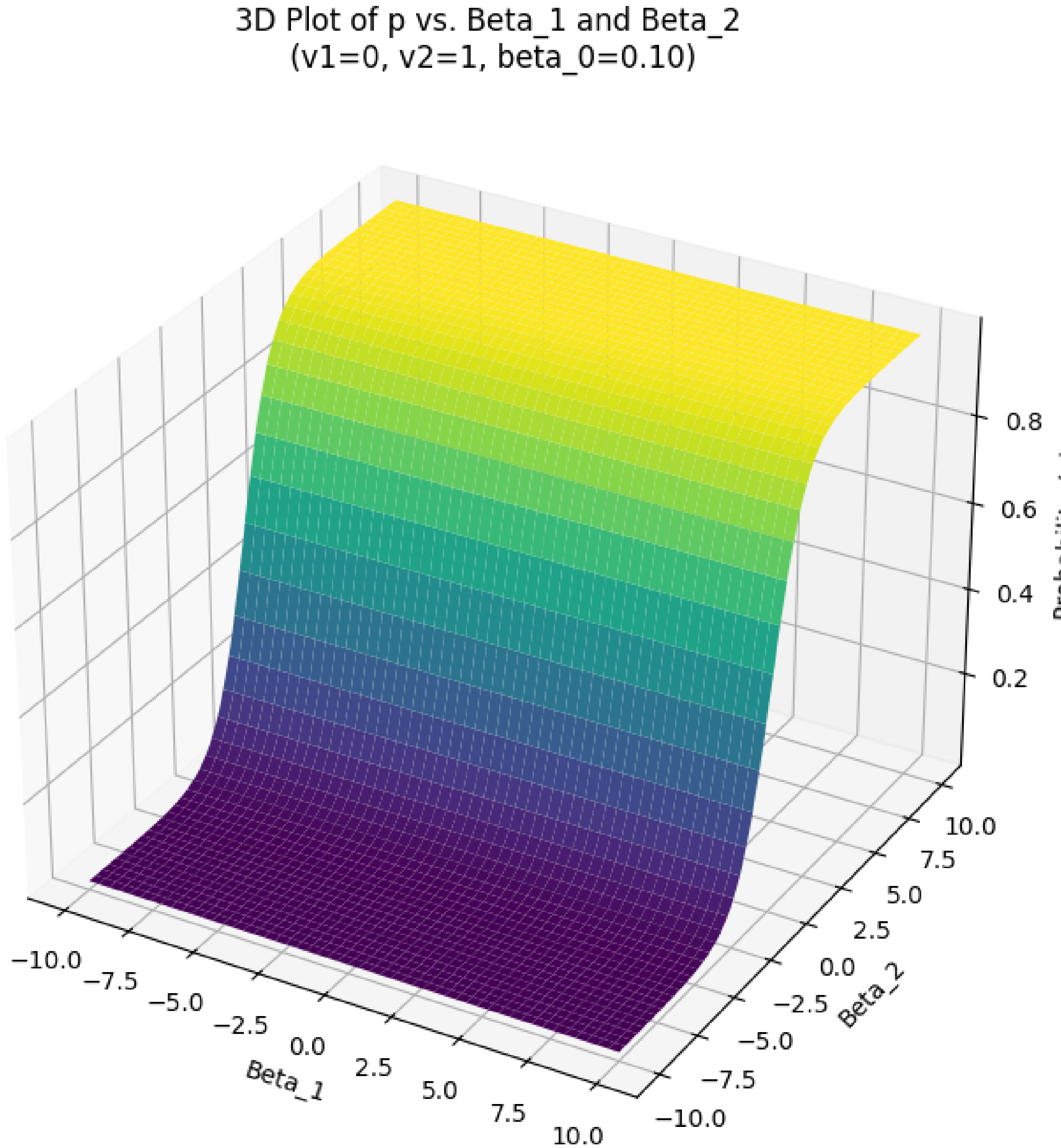
This is the logistic function, and it shows how p, the probability of the event, can be cal
a linear combination of predictor variables through a logistic transformation.

We can use the following code in python to visualize what this looks like when $M = 2$:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Define the logistic function
def logistic(beta_0, beta_1, beta_2, v1, v2):
    z = beta_0 + beta_1 * v1 + beta_2 * v2
    return 1 / (1 + np.exp(-z))

# Define the ranges and values for the plot
beta_0_values = np.linspace(-10, 10, 100)  # Random constant beta_0 from -1 to 1
beta_1_values = np.linspace(-10, 10, 1000)  # Range for beta_1
beta_2_values = np.linspace(-10, 10, 1000)  # Range for beta_2
v1, v2 = np.random.choice([0, 1], 2)  # Random values 0 or 1 for v1 and v2

# Meshgrid for beta_1 and beta_2
```

```
beta_1_grid, beta_2_grid = np.meshgrid(beta_1_values, beta_2_values)

# Calculate probabilities for the grid
p_values = logistic(beta_0_values[50], beta_1_grid, beta_2_grid, v1, v2)
# Using middle value of beta_0

# Create a 3D plot
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(beta_1_grid, beta_2_grid, p_values, cmap='viridis')

# Labels and title
ax.set_xlabel('Beta_1')
ax.set_ylabel('Beta_2')
ax.set_zlabel('Probability (p)')
ax.set_title('3D Plot of p vs. Beta_1 and Beta_2\n (v1={}, v2={}, beta_0={:.2f})

# Color bar
fig.colorbar(surf, ax=ax, shrink=0.5, aspect=5)

plt.show()
```

Voilá:


3D Plot of p vs. Beta_1 and Beta_2
(v1=0, v2=1, beta_0=0.10)
Beta_1
Beta_2
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
0.2
0.4
0.6
0.8