

# How to Easily Plot a Logistic Regression Curve in Python

Authored by  
**stats writer**

December 2, 2025

## RECOMMENDED CITATION

stats writer (2025). *How to Easily Plot a Logistic Regression Curve in Python*.  
PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=103754>

Visualizing statistical models is a critical step in understanding and validating machine learning results. When dealing with classification problems, particularly those involving binary outcomes, the Logistic Regression model is frequently employed. Plotting the characteristic S-shaped curve of this model allows researchers and developers to clearly interpret how changes in an input feature, or predictor variable, influence the probability of the positive outcome.

In Python, the process of plotting a Logistic Regression curve is streamlined and efficient, thanks primarily to powerful libraries dedicated to data visualization. This detailed guide will walk you through leveraging the functionality within the Seaborn library, defining the necessary axes, and plotting the relationship between a continuous feature and the predicted probability, ensuring a clean and insightful graphical representation.

## Understanding Logistic Regression Visualization

Unlike linear regression, which models a straight line relationship, Logistic Regression models the probability of a binary event occurring. This probability function, derived from the logistic function (or sigmoid function), always maps outputs to a range between 0 and 1, resulting in the distinctive S-curve. Plotting this curve provides immediate insight into the decision boundaries and sensitivity of the model across the range of the independent variable.

Effective data visualization is essential for communicating the non-linear relationship inherent in these models. When we plot a logistic curve against raw data points (often scattered along  $y=0$  and  $y=1$ ), we are visually assessing how well the model separates the two classes and where the predicted probability rapidly shifts from one class to the other. This visualization step is crucial for initial model validation and explanatory data analysis.

## Leveraging Seaborn for Statistical Plots

Seaborn is a specialized Python library built on top of Matplotlib, designed to create informative and appealing statistical graphics. It simplifies the complexity often associated with raw Matplotlib commands, especially when fitting and plotting regression models. For plotting a logistic fit, Seaborn includes a dedicated function: `regplot`.

The `regplot` function is highly versatile, primarily designed for fitting and plotting linear relationships between two variables. However, by setting the `logistic` parameter to `True`, we instruct Seaborn to estimate a logistic model based on the provided data and overlay the resulting probability curve onto the scatter plot. This powerful abstraction saves significant time compared to manually calculating probabilities and plotting the curve using lower-level tools.

## Syntax and Essential Parameters of `regplot`

To effectively plot the curve, we must import `Seaborn` and then call the `regplot` function, supplying the key parameters: the independent variable (`x`), the dependent binary variable (`y`), the dataset (`data`), and crucially, the `logistic=True` flag. It is also good practice to suppress the confidence interval calculation, which can sometimes clutter the visualization for simple interpretation, by setting `ci=None`.

The following structure demonstrates the fundamental syntax required to generate the basic Logistic Regression visualization:

```
import seaborn as sns
```

```
sns.regplot(x=x, y=y, data=df, logistic=True, ci=None)
```

This concise code snippet initiates the plotting process. The variables `x`, `y`, and `df` must be correctly defined beforehand, mapping to the column names or series objects representing the predictor variable and the binary response variable within the DataFrame. The practical application of this syntax is demonstrated in the subsequent example using a real-world financial dataset.

### Example: Loading and Exploring the Default Dataset

For a concrete illustration of plotting the logistic curve, we will utilize the publicly available **Default** dataset, frequently referenced in statistical learning texts, specifically the Introduction to Statistical Learning book. This dataset provides rich information suitable for demonstrating binary classification. Before plotting, we must load the data into a Pandas DataFrame and inspect its structure.

The first step involves importing necessary libraries, including Pandas for data manipulation, and loading the CSV file directly from a remote URL. Reviewing the initial rows ensures the data has been imported correctly and provides context regarding the columns we will use for our prediction task.

```
#import dataset from CSV file on Github
```

```
url = "https://raw.githubusercontent.com/arabpsychology/Python-Guides/main/default.csv"
```

```
data = pd.read_csv(url)
```

```
#view first six rows of dataset
```

```
data
```

```
default student balance income
```

```
0 0 0 729.526495 44361.625074
1 0 1 817.180407 12106.134700
2 0 0 1073.549164 31767.138947
3 0 0 529.250605 35704.493935
4 0 0 785.655883 38463.495879
5 0 1 919.588530 7491.558572
```

This dataset, comprising 10,000 observations, contains several features relevant to consumer credit risk assessment. We are specifically interested in using a continuous feature to predict a binary outcome. The key variables within the dataset are defined as follows:

**default:** A binary indicator (0 or 1) specifying whether an individual failed to meet debt obligations. This is our response variable.

**student:** A binary indicator specifying whether the individual is currently enrolled as a student.

**balance:** The average credit card balance carried by the individual.

**income:** The annual income of the individual.

## Generating the Base Logistic Regression Curve

Our objective is to model the probability of **default** based solely on the individual's credit card **balance**. In this model setup, `balance` serves as the predictor variable, and `default` is the outcome we wish to predict. Using the loaded Pandas DataFrame, we isolate these two variables and pass them to the `regplot` function.

Defining the variables explicitly before plotting ensures clarity and repeatability. The `x` variable will hold the continuous feature (`balance`), and `y` will hold the binary response (`default`). By setting `logistic=True`, Seaborn automatically calculates the coefficients for the Logistic Regression model and plots the resulting sigmoid curve over the raw data points.

**#define the predictor variable and the response variable**

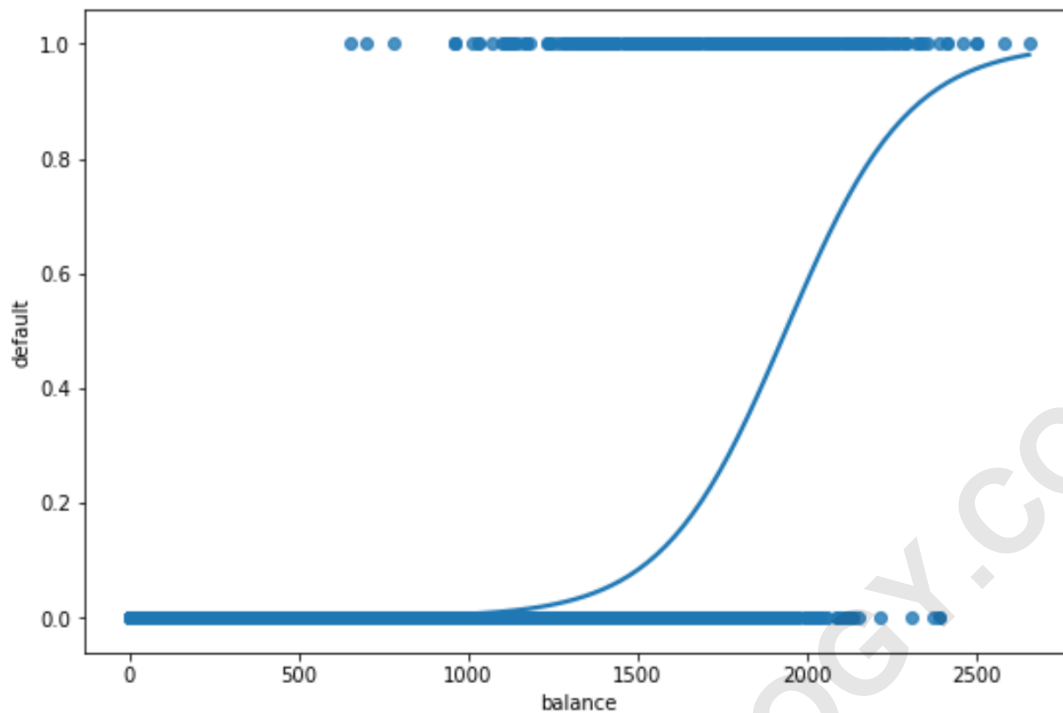
```
x = data
```

```
y = data
```

```
#plot logistic regression curve
```

```
sns.regplot(x=x, y=y, data=data, logistic=True, ci=None)
```

Executing this code generates the fundamental visualization shown below. This plot is essential for visually assessing the strength and nature of the relationship between the credit card balance and the probability of default, offering a crucial step in exploratory data visualization.



## Interpreting the Base Logistic Regression Plot

In the resulting visualization, the horizontal axis (x-axis) represents the values of the predictor variable, which is the credit card "balance." The vertical axis (y-axis) represents the predicted probability of the positive outcome, which in this case is the probability of "defaulting." The raw data points are plotted at  $y=0$  (no default) and  $y=1$  (default), showing the actual outcomes for each balance level.

The smooth S-shaped curve indicates the model's prediction of probability as the balance increases. A steep incline in the middle of the curve signifies that a small increase in balance at that range leads to a rapid increase in the predicted probability of default. Conversely, flat regions at the extremes suggest that changes in balance have less impact on probability. Observing the plot, it is immediately clear that higher values of credit card balance are strongly associated with significantly higher predicted probabilities that an individual will default on their debt. This visual evidence reinforces the statistical findings of the underlying Logistic Regression model.

## Customizing the Visualization for Enhanced Clarity

While the default plot provides necessary statistical information, customizing the aesthetic elements can significantly improve readability and impact. Seaborn's `regplot` function allows for detailed customization of the scatter points and the regression line itself through specific keyword arguments. These arguments are passed via dictionaries: `scatter_kws` for the data points and

`line_kws` for the fitted curve.

Using these keyword arguments, we can easily modify the color, size, and style of the markers and the line. For instance, to ensure maximum contrast and clear distinction between the raw data and the model fit, we might choose to plot the data points in black and the regression curve in a highly visible color like red. This customization is crucial for generating publication-quality figures or presentations that require specific branding or accessibility standards.

### #define the predictor variable and the response variable

```
x = data
```

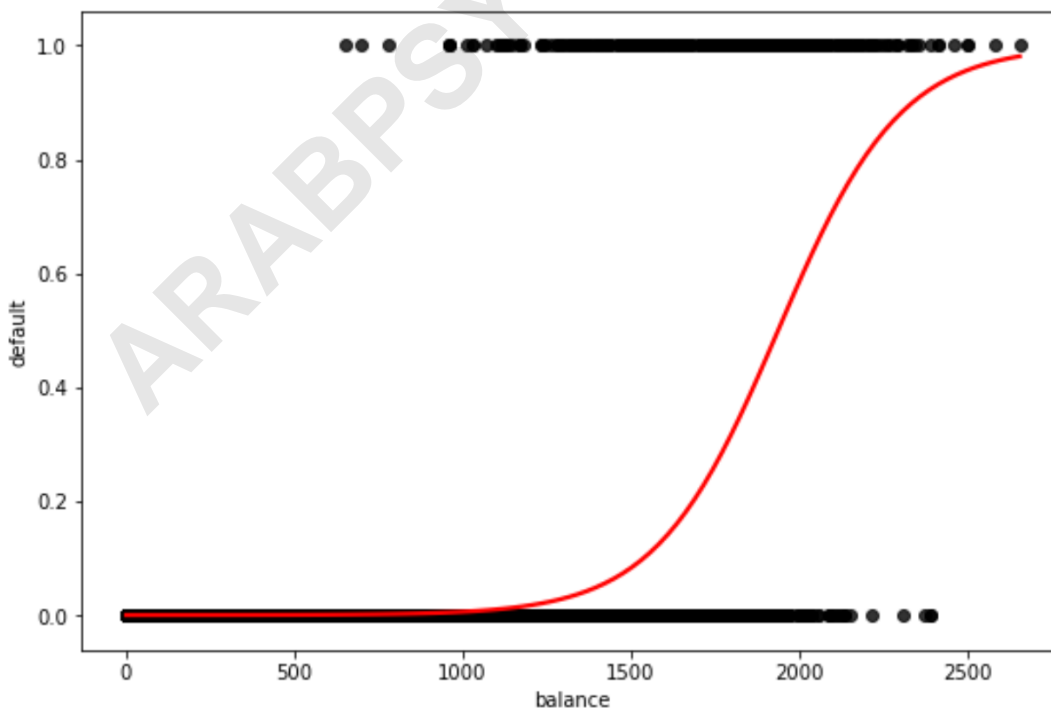
```
y = data
```

```
#plot logistic regression curve with black points and red line
```

```
sns.regplot(x=x, y=y, data=data, logistic=True, ci=None),
```

```
scatter_kws={'color': 'black'}, line_kws={'color': 'red'})
```

By including `scatter_kws` and `line_kws` dictionaries, we override the default color scheme, resulting in a clearer separation between the observed data (black points) and the predicted relationship (red curve), as displayed in the resulting plot below. This ability to fine-tune the visualization is a core strength of using [Pandas](#) and [Seaborn](#) together for statistical analysis and [data visualization](#).



## Conclusion and Further Exploration

Plotting the predictor variable against the predicted probability using Pandas and Seaborn's `regplot` function is an indispensable technique for exploring binary classification models. This technique provides immediate visual confirmation of the model's performance and the directional influence of independent features on the outcome. The simplicity of the code, combined with the statistical power of the visualization, makes this method highly accessible for Python users engaged in predictive modeling.

Mastering these visualization techniques ensures that the insights derived from complex statistical modeling are communicated effectively and accurately. Users are encouraged to experiment with different customization options, such as adjusting marker size or line width, to optimize the visual representation for their specific audience and publication needs.

The following tutorials provide additional information about logistic regression and related plotting techniques:

[Introduction to Statistical Modeling](#)

[Advanced Machine Learning Visualization](#)