# Learning Logistic Regression with Statsmodels in Python

Authored by

**Mohammed looti**

October 28, 2025

## Introduction to Logistic Regression and Statsmodels

Welcome to this detailed guide focused on implementing **logistic regression**, a cornerstone method in predictive analytics, using the highly regarded **Statsmodels** library within the **Python** ecosystem. Unlike traditional linear regression, logistic regression is specifically designed for modeling the probability of a binary or categorical outcome. It is indispensable when the goal is to predict the likelihood of an event belonging to one of two classes, such as predicting customer churn (yes/no), loan default (default/no default), or, as in our example, student success (pass/fail). The core strength of this model lies in its use of the logistic function (or sigmoid function) to transform outcomes into probabilities, ensuring results are bounded between 0 and 1.

The **Statsmodels** package stands out as an exceptional tool for rigorous statistical analysis. It offers a comprehensive suite of classes and functions tailored for fitting various **statistical models**, executing hypothesis tests, and generating detailed statistical reports. While many data science libraries prioritize predictive speed, Statsmodels emphasizes statistical inference and interpretability, making it the preferred choice for researchers, academics, and professionals who require deep understanding of model parameters and their significance. Its structure closely mirrors the formulaic approach common in R, simplifying the specification of complex models involving interactions or categorical variables.

Throughout this step-by-step tutorial, we will execute a complete practical example, guiding you through every phase: from initial data structuring to final model interpretation and evaluation. By the conclusion of this guide, you will be proficient in leveraging Statsmodels to not only build a robust logistic regression model but also to accurately interpret its coefficients, assess predictor significance, and understand crucial performance metrics like the Pseudo R-squared and the Likelihood Ratio Test (LLR). This foundational knowledge will empower you to apply these sophisticated analytical techniques to diverse real-world datasets.

## Step 1: Structuring Data for Statistical Modeling with Pandas

The foundation of any successful statistical analysis is well-prepared data. Our first essential task involves constructing and organizing the dataset using the powerful data manipulation library, Pandas. We will specifically utilize a **Pandas DataFrame**, which serves as the primary two-dimensional, labeled data structure in Python, perfectly suited for tabular data required for regression analysis.

Our illustrative dataset is designed to predict student exam performance based on two independent variables. The variables in this DataFrame are defined precisely for our logistic regression analysis:

**Hours Studied:** A quantitative variable, measured as an integer, representing the total study time

invested by the student. This acts as our continuous predictor variable.
**Study Method:** A qualitative or categorical variable, distinguishing between two distinct study methodologies, 'Method A' and 'Method B'. Logistic regression handles such categorical predictors by converting them into dummy variables automatically, typically using one level as the baseline reference.
**Exam Result:** Our critical binary dependent variable, coded as 'Pass' (represented by 1) or 'Fail' (represented by 0). The model will estimate the probability of the outcome being 1 (Pass).

The primary goal is to establish a statistical relationship using the **logistic regression model**, where 'Exam Result' is predicted by a combination of the 'Hours Studied' (continuous) and the 'Study Method' (categorical). The following Python code snippet initiates the creation of this sample **DataFrame**, populates it with synthetic but representative data points, and verifies its structure by printing the initial rows, confirming that our data is ready for the modeling phase.

**import pandas as pd**

#create DataFrame
df = pd.DataFrame({'result': ,
'hours': ,
'method': })

#view first five rows of DataFrame
df.head()

result hours method
0 0 1 A
1 1 2 A
2 0 2 A
3 0 2 B
4 0 3 B

## Step 2: Fitting the Logistic Regression Model with Statsmodels Formula API

Once the data is structured correctly within the Pandas DataFrame, the next step is the actual implementation and fitting of the statistical model. **Statsmodels** streamlines this process significantly through its dedicated formula API, which allows users to define the model structure using R-style formula notation. This is often more intuitive for statistical modeling than specifying matrices of predictors.

We import the necessary functionality using `statsmodels.formula.api` and utilize the specific function for binary outcomes: **`logit()`**. The model is specified via a formula string, `'result ~`

`hours + method'`. Here, `result` is the dependent variable (the outcome we are trying to predict), and `hours` and `method` are the independent predictor variables. The formula clearly communicates that the result is modeled as a function of the hours studied and the study method employed.

The defined model object is then estimated by invoking the **`.fit()`** method. This step employs Maximum Likelihood Estimation (MLE), the standard technique for fitting logistic regression, which iteratively seeks the parameter values (coefficients) that maximize the probability of observing the actual outcome data. Upon successful convergence of the optimization algorithm, the resulting fitted model object contains all the calculated statistics, which we then output using the `.summary()` method to gain comprehensive insight into our model's performance and parameter estimates.

**import statsmodels.formula.api as smf**

#fit logistic regression model
model = smf.logit('result ~ hours + method', data=df).fit()

#view model summary
print(model.summary())

Optimization terminated successfully.
Current function value: 0.557786
Iterations 5
Logit Regression Results
==========
===
Dep. Variable: result No. Observations: 20
Model: Logit Df Residuals: 17
Method: MLE Df Model: 2
Date: Mon, 22 Aug 2022 Pseudo R-squ.: 0.1894
Time: 09:53:35 Log-Likelihood: -11.156
converged: True LL-Null: -13.763
Covariance Type: nonrobust LLR p-value: 0.07375
==========
====
coef std err z P>|z|
---
Intercept -2.1569 1.416 -1.523 0.128 -4.932 0.618
method 0.0875 1.051 0.083 0.934 -1.973 2.148
hours 0.4909 0.245 2.002 0.045 0.010 0.972
==========

====

## Step 3: Interpreting Estimated Coefficients and Statistical Significance

The model summary generated by Statsmodels is the central output for statistical inference. We must carefully analyze the table of **coefficients** to understand how each independent variable influences the dependent variable. In logistic regression, the `coef` column provides the estimated parameter values that quantify the change in the **log odds** of the outcome (passing the exam) for a one-unit change in the predictor, assuming all other variables are held constant. The log odds, or logit, is the natural logarithm of the odds ratio, serving as the linear component of the logistic model.

Let's break down the meaning of the estimated **coefficients** in our output:

For `method`, the coefficient is `0.0875`. Since 'Method A' is the baseline (reference) category, this positive coefficient suggests that switching from 'Method A' to 'Method B' marginally increases the estimated **log odds** of passing the exam by 0.0875, holding study hours constant.
For the continuous variable `hours`, the coefficient is `0.4909`. This positive value indicates a substantial relationship: for every additional hour a student studies, the estimated **log odds** of achieving a passing result increases by 0.4909, provided the study method remains the same. This suggests that study duration is a key driver of success.

Crucial to evaluating these relationships is the `P>|z|` column, which lists the **p-values** derived from the Wald z-test for each coefficient. The p-value assesses the probability of observing a coefficient as extreme as the one estimated, assuming the null hypothesis (that the true coefficient is zero and there is no relationship) is true. We use these p-values to determine the **statistical significance** of each predictor, typically against an alpha level of 0.05.

The p-value for `method` is `0.934`. Given that this value far exceeds the standard 0.05 threshold, we fail to reject the null hypothesis. We must conclude that, based on this model and dataset, the choice between 'Method A' and 'Method B' does not have a **statistically significant relationship** with the likelihood of passing the exam.
Conversely, the p-value for `hours` is `0.045`. Since 0.045 is less than 0.05, we reject the null hypothesis. This strongly affirms that the number of hours dedicated to studying is a **statistically significant predictor** of the exam outcome.

## Step 4: Evaluating Overall Model Performance Metrics

While individual coefficient significance is important, we must also evaluate the overall fit and predictive utility of the model. The Statsmodels summary provides several diagnostics that help us

gauge the quality of our **logistic regression model**. We will focus on two key global measures: the Pseudo R-squared and the LLR p-value, which offer insights into the model's explanatory power relative to a baseline.

### 1. Pseudo R-Squared

In the context of logistic regression, the traditional R-squared used in **linear regression** is inappropriate because the model is fitted using Maximum Likelihood Estimation, not least squares. Instead, we rely on Pseudo R-squared metrics, such as McFadden's R-squared (the default in Statsmodels), which attempt to approximate the proportion of variance explained. This metric compares the **log-likelihood** of the fitted model to the log-likelihood of the **null model** (the model containing only the intercept).

Pseudo R-squared values typically range from 0 to 1, but interpretation differs from linear R-squared; even values between 0.2 and 0.4 often signify a very good fit in certain application fields. A higher value suggests that the predictors included in the model contribute significantly to explaining the variability in the outcome. In our current analysis, the Pseudo R-squared is calculated as `0.1894`. This modest value suggests that while the predictors (especially study hours) are useful, they only account for a small to moderate amount of the variance in the exam result, implying that other unmeasured factors likely play a substantial role in student performance.

### 2. LLR p-value (Likelihood Ratio Test p-value)

The Likelihood Ratio Test (LRT) is a fundamental statistical procedure used here to compare the fit of our full model against the **null model**. The LLR p-value, which is the result of the **Likelihood Ratio Test**, acts as a comprehensive overall measure of model utility, similar to the F-test in **linear regression**.

The null hypothesis for the LRT is that all regression **coefficients** (excluding the intercept) are simultaneously equal to zero. If the LLR p-value is less than our predefined **significance level** (alpha), we reject this null hypothesis, concluding that the full model provides a statistically superior fit to the data compared to having no predictors at all. In our output, the LLR p-value is `0.07375`. If we strictly maintain the common significance level of $\alpha = 0.05$, this result is marginally non-significant, suggesting we cannot definitively prove the overall model is better than the null. However, if the context permits a slightly more lenient threshold, such as $\alpha = 0.10$, the model would then be deemed statistically significant. This border-line result emphasizes the importance of context and domain knowledge in defining the appropriate alpha level.

## Conclusion and Final Thoughts

This tutorial provided a thorough exploration of implementing **logistic regression** using the inferential power of **Statsmodels** within the **Python** programming environment. We successfully navigated the entire analytical pipeline, starting with constructing a robust **Pandas DataFrame**, specifying and fitting the model using the intuitive `logit()` function, and critically interpreting the resulting statistical summary.

We gained valuable insights by examining the **coefficients** (which quantify the impact on log odds), the **p-values** (which confirm that study hours, but not the study method, significantly predict the outcome), and global metrics like the Pseudo R-squared and LLR p-value. These tools collectively allow us to assess both the specific influence of predictors and the overall explanatory capacity of the model. While our sample model demonstrated that study hours are a statistically significant factor, the overall model fit was moderate, highlighting the existence of unmodeled complexity in the data.

As a powerful technique for binary classification problems, logistic regression, when implemented via Statsmodels, offers a statistically transparent and rigorous framework. We encourage you to utilize this methodology in your own data analysis tasks, always maintaining a critical eye on model assumptions, carefully interpreting the results within the domain context, and continuously seeking to improve predictive power through feature engineering and data refinement.

## Additional Resources for Further Learning

To continue advancing your skills in statistical analysis and machine learning using **Python**, the following curated resources are highly recommended. They provide detailed explorations into essential tools and advanced topics, serving as excellent complements to the foundational concepts covered in this tutorial.