4/28/2025

Machine Learning for Pricing

Technical Introduction Using Logistic Regression

Purpose:

Provide a primer on applied machine learning in the context of our unique business case. After studying this document, readers will possess a working knowledge of how to regress a logistic function onto data sourced from the [*redacted*] warehouse in order to model customer price sensitivity.

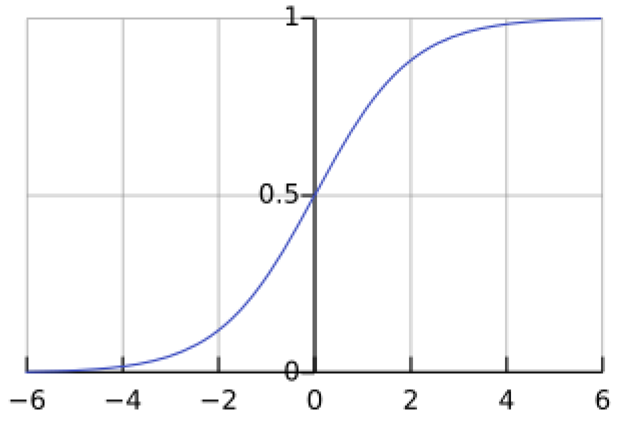
Part 1: Introductory Concepts

The Logistic Function (aka the Sigmoid Curve)

The logistic function is a mathematical equation which was first introduced by Pierre François Verhulst in 1838 in order to model population growth. With its unique S-shape, it progresses in three distinct stages: initially, growing exponentially; then transitioning into slower, more linear growth as saturation begins; and finally, at maturity, an exponentially decaying approach to its maximum value (essentially the reverse of how it began). This was found to be a good function for modelling population growth, but over time attained wide usefulness in representing phenomena in a number of fields, one of which was probability. Today, it is one of the most widely used models in applied machine learning and often serves as the model of choice for the first iteration of a concept, before moving onto more complex models like gradient boosters. Its purpose is to model what are known as “binary classification problems” (where the goal is to predict the probability of one of two outcomes).

$$\boxed{W(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}}$$

The particular form of the logistic function which is used in statistics to model probability is found in the equation to the right. The numerator represents the maximum value (in this case, 1 or 100% probability), while the denominator is comprised of a number that will always be greater than 1 (therefore never allowing the equation to reach its maximum value, 1). Within the constant e’s exponent, there are three key terms: B_0 represents the intercept (midpoint) of the function, B_1 represents the slope, and x is the independent variable which drives the function. The dependent variable (which we have labelled W(x) in order to represent customer win probability) is the probabilistic value which the function generates across a range of x inputs (i.e. potential hourly rates). It will always be a number between 0 and 1.



The Process of Regression

In applied machine learning, one word that is very important to understand is “regression”. To “regress” is essentially to “fit” – that is, to fit a function or a model onto your dataset. Regression is the process of adjusting the coefficients or “weights” of your model until its behavior conforms to the shape of your data. In the case of a logistic function, what this would look like is testing out thousands of combinations of B_0 and B_1 until you find the combination that best fits the sigmoid curve onto your dataset. Once you have the “weights” that make the function best fit your data, you now have a personalized logistic function which in theory should be able to predict the probability of future events based on hypothetical input values (x).

The Lynchpin of the Regression Process: The Cost Function

We are now getting into the nuts and bolts of the essential process which undergirds all of machine learning: fitting the model to your data! This is commonly referred to as “training” the model. The training process is very simple: you essentially test many different versions of your function on your dataset (in other words, different combinations of B_0 and B_1), observing what probabilities they output for each x value input. From there, you “grade” each version of the function by how close to or far from the true results its output was. In the end, you choose the version of the function which predicted values that were closest to your dataset on average.

A vital question, however, is this: how do you “grade” the model’s performance? What is the measure of how fair or poor its performance is in comparison to the actual outcomes in your dataset? In machine learning, this measure of error is known as the “cost function” – and each type of model has its own unique cost function, or standard for measuring error. In the case of logistic regression, we use what is known as “log-loss.” Log-loss, also known as “cross-entropy loss”, is a logarithmic transformation of the likelihood function. It is a specialized method of measuring the difference between predicted probability and actual outcomes, which penalizes errors more severely than it rewards conformity, and is the gold standard for assessing the fit of a logistic function to real data. Its formula is as follows, where P represents the actual outcome and p represents your model’s predicted probability:

$$\text{Log-Loss}(P, p) = -(P \log(p) + (1 - P) \log(1 - p))$$

To tie it all together, the process of training a logistic regression model consists of testing out thousands of different combinations of coefficients B_0 and B_1 (aka “weights”) and seeing how they perform against the real dataset by calculating the log-loss value of each predicted outcome against the actual outcomes in the data. For each different version of the function, you look at the sum of the log-losses over the entire dataset, and move toward the function with the lowest total log-loss (most accurate fit). Eventually, you find a combination of weights which yields the closest fit which is possible to your dataset given the variables or parameters you have chosen (there can be more than one “x”, or independent variable…but we’ll save that for later, let’s stick with one for now).

We now have the foundational concepts of machine learning in place and are ready to practice manually regressing a logistic function onto a sample dataset in Excel.

Part 2: Manual Regression in Excel

Composing the Required Columns

Before we jump into Python and begin to use the automated models which are available to us in libraries like Sci-kit Learn, it is very helpful for the learning process to conduct the regression process "manually" in an Excel spreadsheet. This really helps to cement the concepts discussed in Part 1 and to see them in action, before we outsource all of that work to a pre-built model.

To begin, then, we will assume you are working with the win/loss data for the handyman trade which is found in [*redacted*] database. Save an Excel file with this data downloaded from [*redacted*], and add the following columns to the data:

1. Win/loss
    - This should be a column which displays a 0 for all bids marked as lost and a 1 for all bids marked as won
2. Predicted Probability (p)
3. Log-Loss

The win/loss column should be fairly simple to create using an if statement in Excel. The following two columns, however, will require some more explanation. For the "Predicted Probability" column, we need to write the formula for our logistic function. First, however, we need to create two reference cells: one for B_0, and one for B_1. Put these at the top of your sheet and enter a small value like 0.5 in each cell to start with. Once you've created those, write this formula using your reference cells and drag it down the column for all the rows of your dataset:

$$= 1 / (1 + \mathrm{EXP}(- (B_0 + B_1 * x)))$$

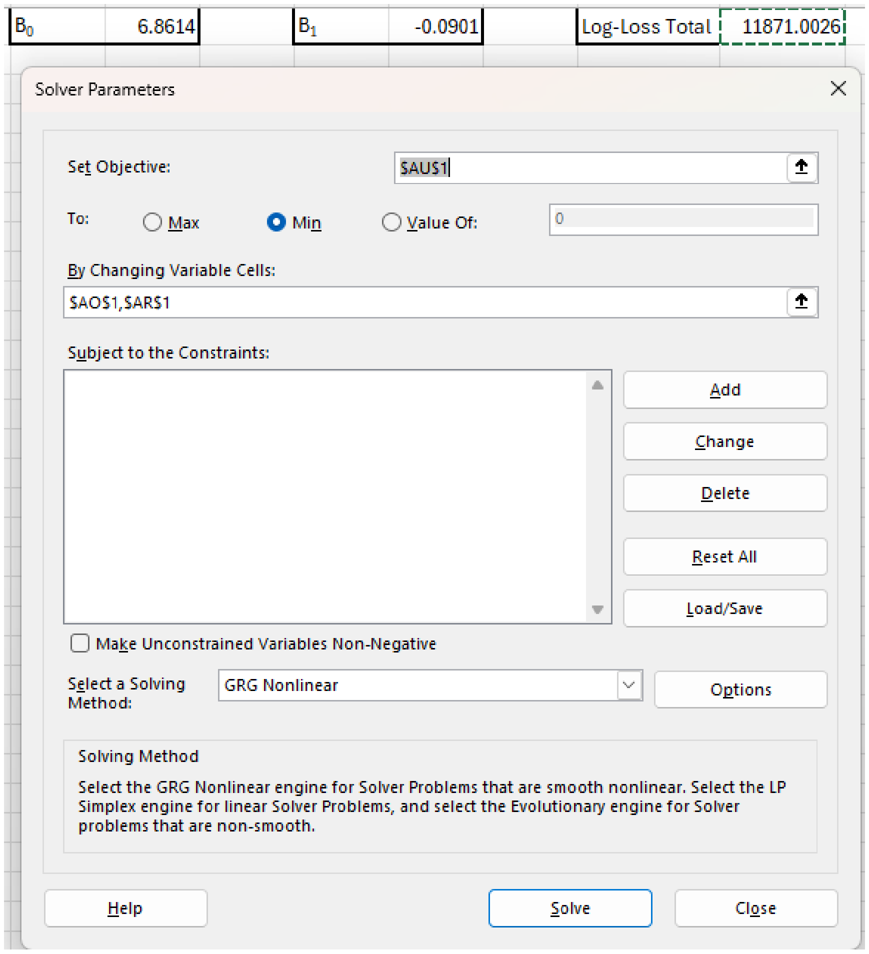
In this formula, x is the customer hourly rate that was presented to the customer for each bid. Link the formula to that cell for each row. Once you have written this formula and dragged it down your dataset, you have a logistic function which is attempting to model your data! Now you're ready to build out the final piece: the log-loss column. For this, you will write the following formula and copy it down each row:

$$= - (P * \mathrm{LN}(p) + (1 - P) * \mathrm{LN}(1 - p))$$

In this formula, P is the actual outcome from your win/loss column (either 0 or 1) and p is the predicted probability from the column we just built above. Once you have this column finished, create two more reference cells at the top of your sheet next to B_0 and B_1 called "Log-Loss Total" and "Avg Log-Loss", where you sum and average all the log-losses from your dataset. These will be crucial in a moment.

Implementing Excel Solver

We have now arrived at the really exciting step in the process: regressing a function onto our dataset! The setup is complete and now we can start testing different weights to arrive at a function which is personally tailored to our dataset. Thankfully, however, we do not have to manually test tens of thousands of different combinations of coefficients to arrive at a fit – Excel actually has an in-house feature that will do this for us in seconds, which is called “Solver”. In order to make it available in your Excel ribbon, you will have to navigate to File->Options->Add-ins->Manage: Excel Add-ins->Click go, check Solver Add-in, click OK. Once completed, you will find it in the “Data” section of your ribbon, all the way to the right in the “Analyze” group. Click on it, then mirror the setup in the screen clipping below and click “Solve”.



B_0	6.8614	B_1	-0.0901	Log-Loss Total	11871.0026

Solver Parameters

Set Objective: AU1

To: ( ) Max [✓ Min] ( ) Value Of: 0

By Changing Variable Cells: AO1,AR1

Subject to the Constraints:

☐ Make Unconstrained Variables Non-Negative

Select a Solving Method: GRG Nonlinear

Solving Method

Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

Solver will now take around 10-15 seconds trying many different combinations of your coefficients, looking for the combination which best minimizes your Log-Loss Total. As it narrows down to the best possible fit to your data, it will eventually trigger a prompt box which explains it has finished. And **at this point, you have successfully trained your first machine learning model!** Stored at the top of your sheet in your reference cells for B_0 and B_1 are a precious treasure: the “weights” to your dataset. These are the keys to unlocking the meaning within your data, and capturing the story it’s telling in order to predict the outcomes of new bids in the future. These weights are the coefficients that define your specific logistic function which is tailor-fitted to the handyman win/loss dataset, and they are the outcome of the training exercise you’ve just gone through. These weights are the distilled, compressed wisdom of all your data, analyzed and interpreted. And here’s a quick secret, if you didn’t already know: this process is the same basic framework for all of machine learning, even for the extremely complex neural networks that are at the cutting edge of AI research and development, known as “deep learning models”. Those companies, just like us, are training models on data, using a cost function to measure error, iterating to reduce the cost function as far as attainable, and then saving the weights which result from the process. These weights are closely guarded by AI companies, and are stored in multi-terabyte files on highly secure servers. Somewhere, on a top secret server, ChatGPT’s weights exist – which is essentially its brain, or the condensed wisdom of all the text and image material it has been trained on. **You have now conducted – on a basic level – the same process which was used to build ChatGPT 4o**.

Part 3: Migrating to Python – The Industry Standard

To Be Continued…