Understanding Overfitting in Machine Learning: Concepts and Examples

Authored by

Mohammed looti

November 6, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *Understanding Overfitting in Machine Learning: Concepts and Examples*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=11857

In the complex and rapidly evolving field of Machine Learning, the primary objective is to construct models that are capable of making accurate and reliable predictions concerning future, unseen data points. We seek not merely to describe existing data, but to derive underlying, generalizable patterns from it.

Consider a practical scenario: we intend to develop a regression model designed to predict the standardized test scores (such as the ACT score) of high school students based on a single predictor variable--the total number of **hours spent studying**. This is a typical supervised learning task where we aim to map inputs to continuous outputs.

To properly construct and calibrate this model, we must first gather a substantial dataset. This dataset will comprise paired observations of study hours and corresponding ACT scores for hundreds of students within a specific population, such as a school district. This collected data forms the critical foundation used to **train** our model.

The training process involves iteratively adjusting the model's internal parameters until it minimizes the discrepancy between its predictions and the actual observed outcomes. Once trained, the model should ideally possess the predictive capability to estimate the score a new student will receive based solely on their reported study hours.

Understanding Mean Squared Error (MSE) and Model Fitness

Evaluating the usefulness and efficacy of any predictive model requires a quantitative measure of performance, commonly referred to as a loss function or cost function. This metric allows us to quantify how closely the model's predictions align with the observed real-world data points. For regression tasks, one of the most widely accepted and frequently utilized metrics is the Mean Squared Error (MSE).

The MSE provides a clear measure of the average squared difference between the estimated values and the actual values. By squaring the error, the metric heavily penalizes predictions that are significantly far off from the true observation, ensuring that large errors have a disproportionate impact on the overall cost calculation. This emphasis on reducing outliers makes MSE a robust choice for optimizing model parameters during the training phase.

The calculation for the Mean Squared Error is formally defined as:

$$MSE = (1/n)*\Sigma(yi - f(xi))2$$

where the variables represent:

n: The total count of observations in the dataset being evaluated.
yi: The actual, observed response value for the ith data point.

f(xi): The predicted response value generated by the model for the ith observation, given its input features.

In practice, the closer the model's computed predictions are to the true observed values, the smaller the resulting MSE will be. A successful model typically strives for the minimum achievable MSE on the data used for evaluation, whether it be the training set or, more importantly, the validation set.

Defining and Identifying the Phenomenon of Overfitting

A prevalent and potentially catastrophic error in the Machine Learning development lifecycle is the optimization of a model purely to minimize the **training MSE**. While a low training error seems desirable, this single-minded focus often causes the model to over-accommodate the specific nuances and idiosyncrasies of the training data itself.

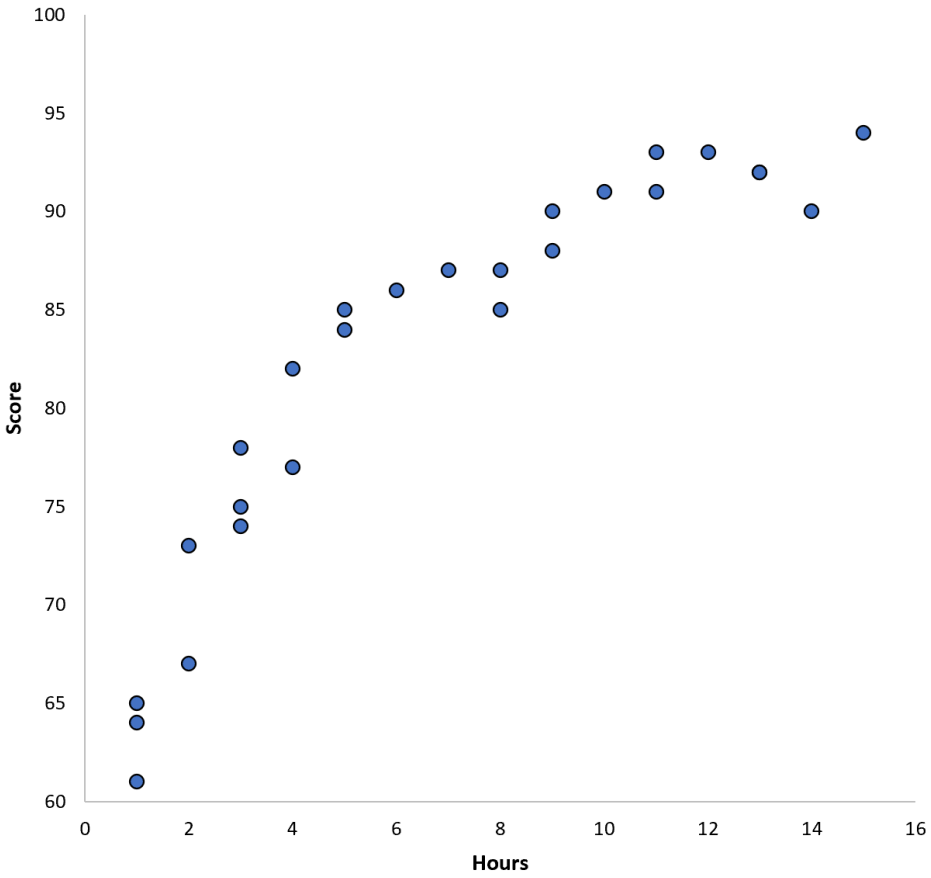
When a model expends too much effort reducing the training error, it begins to learn patterns that are not representative of the true underlying relationship between the variables, but are merely artifacts of **random noise** or statistical anomalies present only in that specific dataset. This process is akin to memorizing answers rather than understanding the concepts.

This critical failure to generalize is known as **overfitting**. It occurs when we "fit" the model too closely or too perfectly to the training examples. Consequently, the resulting model becomes overly complex and loses its utility for its intended purpose: making reliable predictions about new, previously unseen data points. An overfit model exhibits high variance, meaning it is extremely sensitive to minor fluctuations in the input data.

Illustrative Example of Overfitting: Polynomial Regression

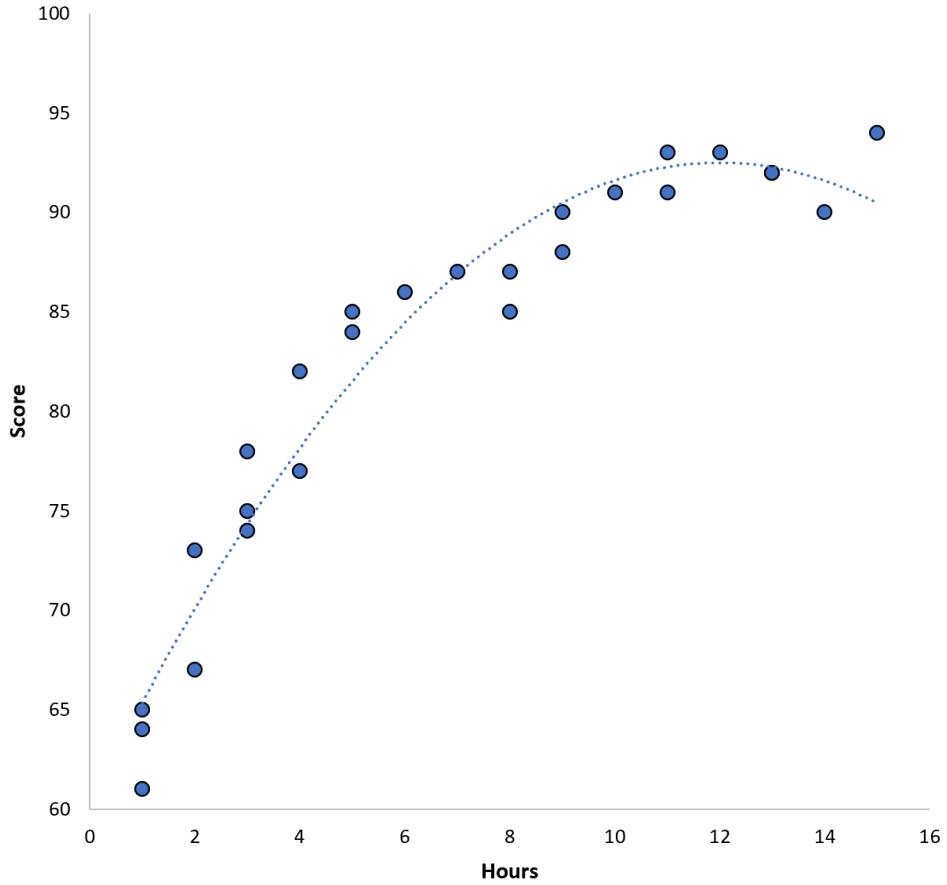
To concretely understand the consequences of overfitting, let us revisit our example of creating a regression model that maps *hours spent studying* to the predicted *ACT score*. Suppose we collect data for 100 students and generate a scatterplot to visually inspect the relationship between the two variables.

The initial visual assessment often suggests a certain functional form. Based on the scatter of the points, the relationship appears to follow a curved, non-linear pattern, perhaps best described by a quadratic function. We proceed by fitting a simpler, second-order quadratic regression model to this training data:



The resulting quadratic model provides a visually reasonable fit to the data, capturing the general upward curve without being unduly influenced by every single data point.

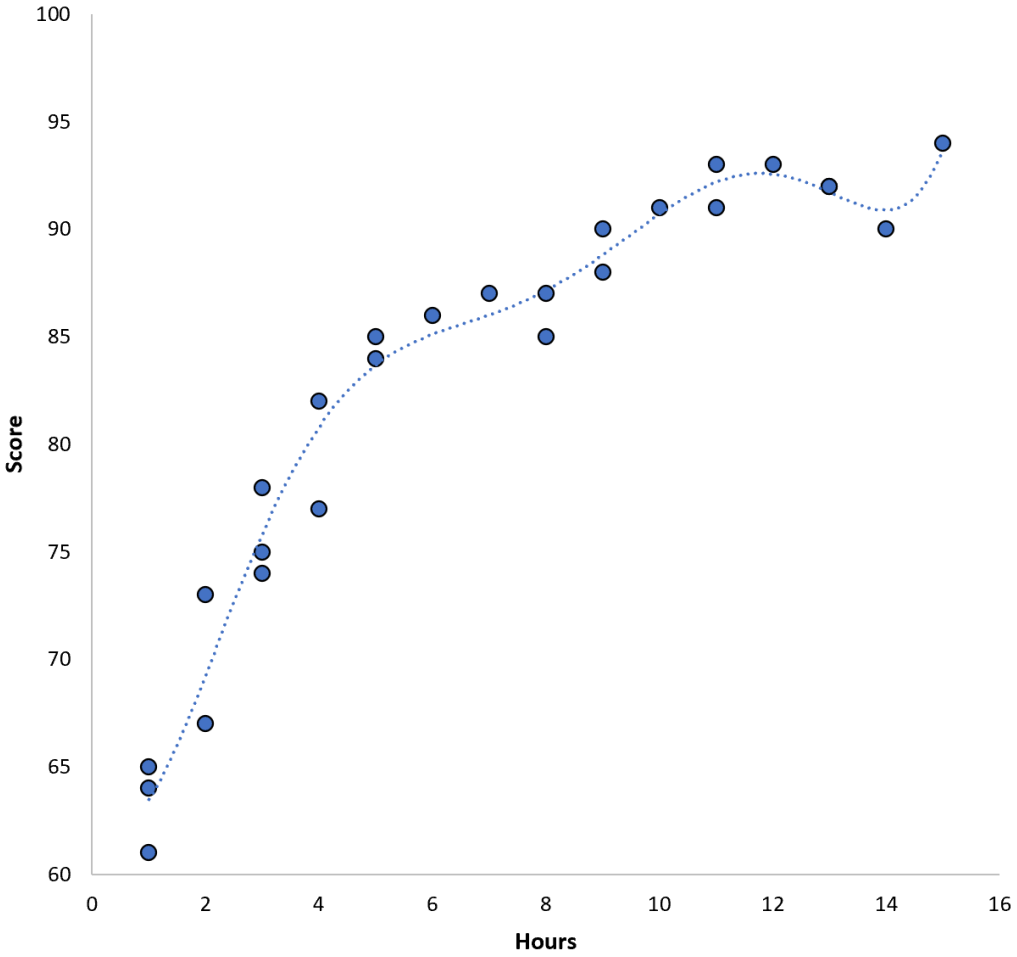
Upon evaluating this initial model, we find that it yields a training Mean Squared Error (MSE) of **3.45**. This figure represents the average squared discrepancy between the model's predictions and the actual ACT scores in our training set. This is a good starting point, showing a reasonable balance between complexity and error.



However, driven by the desire for a lower error score, a developer might attempt to drastically reduce this training MSE by increasing the model's complexity--for instance, by fitting a much higher-order polynomial model, such as a sixth-order polynomial. This significantly increases the number of parameters the model must learn, allowing it to conform much more closely to the specific training points:

The highly complex model equation might look like this:

Score = 64.3 - 7.1*(Hours) + 8.1*(Hours)2 - 2.1*(Hours)3 + 0.2*(Hours)4 - 0.1*(Hours)5 + 0.2(Hours)6



Observing the graph above, it is evident that the regression line for this high-order polynomial model dramatically contorts itself to "hug" every single data point, including those that are clearly outliers or noise. This extreme fidelity to the training data results in a greatly reduced training MSE of just **0.89**, a score much smaller than the 3.45 achieved by the simpler quadratic model. While this looks like an improvement on paper, it signals severe overfitting.

The Critical Distinction: Training MSE vs. Test MSE

The key insight lost when a model is overfit is the distinction between performance on the known training set and performance on unknown, future data. A low **training MSE** merely confirms that the model has successfully memorized the input data, but it tells us nothing about its ability to generalize. The model has learned the noise inherent in the sample, not the true relationship.

The true measure of a model's value is its **Test MSE**, sometimes referred to as the generalization error. This metric assesses the model's performance when applied to data it has never encountered during training. A high-variance, overfit model will produce wildly erratic predictions when encountering minor variations in new input data, leading to a significantly inflated Test MSE.

In the context of our polynomial example, if we applied the highly complex sixth-order model to a new dataset of students, its erratic, wavy curve would almost certainly lead to much larger prediction errors than the smooth, stable curve of the simpler quadratic model. The high-order model would thus yield a much higher Test MSE, confirming that it is overfit and ultimately less useful for real-world prediction despite its stellar performance on the original training set. Selecting models based solely on low training error is the fundamental mistake that overfitting embodies.

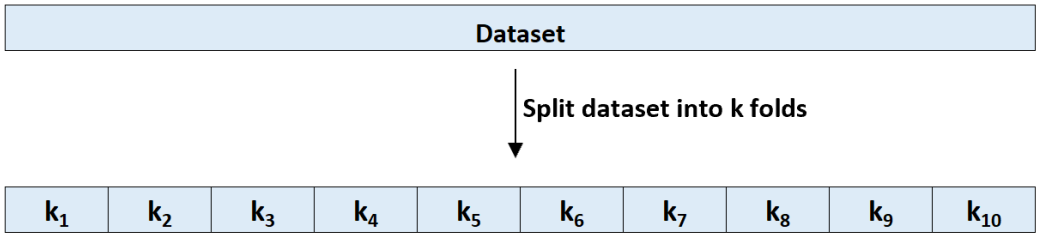
Practical Methods for Detecting and Preventing Overfitting

Since minimizing the generalization error (Test MSE) is the ultimate goal, effective strategies must be employed to accurately estimate this error without using the model's training data. The most robust and commonly accepted method for achieving this is cross-validation. Specifically, **k-fold cross-validation** provides a reliable estimate of how well a model will perform on unseen data by systematically rotating which subset of the data is used for training and which is reserved for testing.

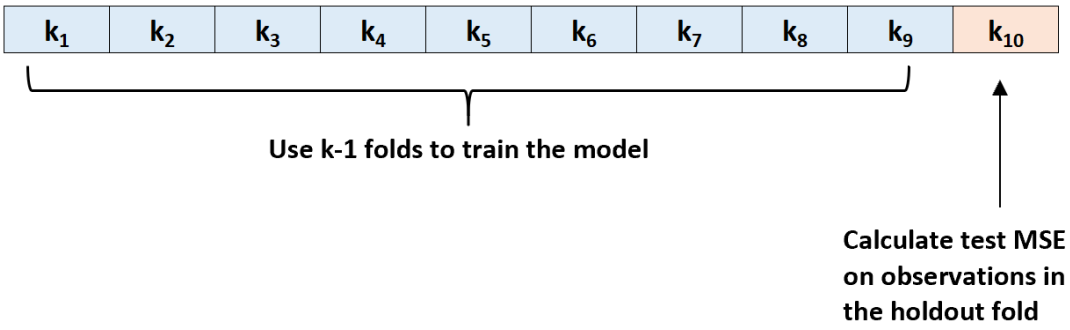
The k-fold cross-validation procedure is a powerful technique designed to ensure that every observation in the dataset is used exactly once for validation. By averaging the results across multiple folds, we obtain a much more stable and unbiased estimate of the model's true predictive capabilities compared to a simple single train/test split.

The typical steps involved in executing k-fold cross-validation are as follows:

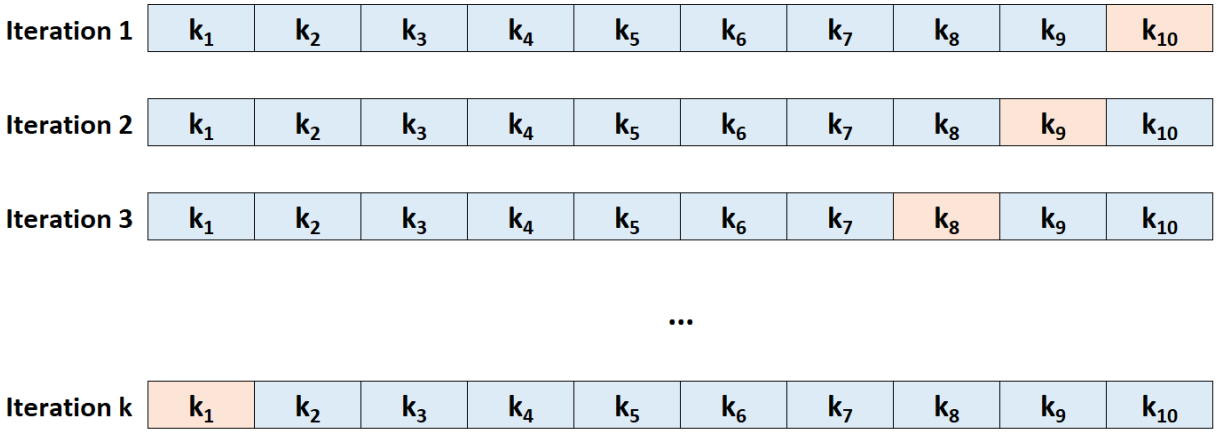
Step 1: Randomly segment the complete dataset into *k* groups, or "folds," ensuring that each fold is of approximately equal size. This division ensures that the testing process covers the entire sample space.



Step 2: Designate one of the folds as the crucial holdout (or validation) set for that iteration. The model is then fit (trained) exclusively on the combined data from the remaining k-1 folds. After training, the Test MSE is calculated solely using the observations within the held-out fold. This provides a measure of generalization error for that specific fold.



Step 3: This entire process is repeated *k* times. In each iteration, a different fold is systematically chosen to serve as the holdout set. This guarantees that every data point participates in the training process (k-1 times) and the testing process (1 time).



Step 4: The overall Test MSE, which serves as our final, unbiased estimate of generalization error, is calculated as the average of the *k* individual Test MSE scores derived from each iteration.

The formula for the final aggregate Test MSE is:

Test MSE = (1/k)*ΣMSEi

where:

k: Represents the chosen number of folds used in the procedure.
MSEi: The Test MSE calculated on the holdout fold during the ith iteration.

By implementing cross-validation, developers can objectively fit and evaluate several different models (e.g., linear, quadratic, cubic, etc.) and accurately compare their average Test MSEs. The

model that yields the lowest average Test MSE is the one chosen for deployment, as it offers the best expectation of performance on future data, successfully mitigating the risk of overfitting and ensuring true generalization.

Additional Resources for Deeper Understanding

What is the Bias-Variance Tradeoff in Machine Learning?
An Introduction to K-Fold Cross-Validation
Regression vs. Classification Models in Machine Learning