Evaluate Models Using Metrics

While debugging a ML model can seem daunting, model metrics show you where to start. The following sections discuss how to evaluate performance using metrics.

Evaluate Quality Using Model Metrics

To evaluate your model's quality, commonly-used metrics are:

- loss (/machine-learning/crash-course/descending-into-ml/training-and-loss)
- accuracy (/machine-learning/crash-course/classification/accuracy)
- precision & recall (/machine-learning/crash-course/classification/precision-and-recall)
- area under the ROC curve (AUC) (/machine-learning/crash-course/classification/roc-and-auc)

For guidance on interpreting these metrics, read the linked content from Machine Learning Crash Content. For additional guidance on specific problems, see the following table.

Problem	Evaluating Quality
Regression	Besides reducing your absolute Mean Square Error (/machine-learning/crash-course/descending-into-ml/training-and-loss) (MSE), reduce your MSE relative to your label values. For example, assume you're predicting prices of two items that have mean prices of 5 and 100. In both cases, assume your MSE is 5. In the first case, the MSE is 100% of your mean price, which is clearly a large error. In the second case, the MSE is 5% of your mean price, which is a reasonable error.
Multiclass classification	If you're predicting a small number of classes, look at per-class metrics individually. When predicting on many classes, you can average the per-class metrics to track overall classification metrics. Alternatively, you can prioritize specific quality goals depending on your needs. For example, if you're classifying objects in images, then you might prioritize the classification quality for people over other objects.

Check Metrics for Important Data Slices

After you have a high-quality model, your model might still perform poorly on subsets of your data. For example, your unicorn predictor must predict well both in the Sahara desert and in New York City, and at all times of the day. However, you have less training data for the Sahara desert. Therefore, you want to track model quality specifically for the Sahara

desert. Such subsets of data, like the subset corresponding to the Sahara desert, are called **data slices**. You should separately monitor data slices where performance is especially important or where your model might perform poorly.

Use your understanding of the data to identify data slices of interest. Then compare model metrics for data slices against the metrics for your entire data set. Checking that your model performs across all data slices helps remove bias. For more, see Fairness: Evaluating for Bias (/machine-learning/crash-course/fairness/evaluating-for-bias).

Use Real-World Metrics

Model metrics do not necessarily measure the real-world impact of your model. For example, you might change a hyperparameter and increase your AUC, but how did the change affect user experience? To measure real-world impact, you need to define separate metrics. For example, you could survey users who see a unicorn appearance prediction to check whether or not they saw a unicorn. Measuring real-world impact helps compare the quality of different iterations of your model.

Previous
← Interpreting Loss Curves (/machine-learning/testing-debugging/metrics/interpretic)
Next
Check Your Understanding →
(/machine-learning/testing-debugging/metrics/check-your-understanding)