MasterfulAI

Benchmark Report

CIFAR-10

Overview

This report shows the forecasted impact of implementing Masterful to improve your model performance, according to standard classification metrics. It also breaks out the specific training techniques within the Masterful platform that could most improve various aspects of your model. The included charts are automatically generated and viewable in the Masterful front-end.

Table of Contents

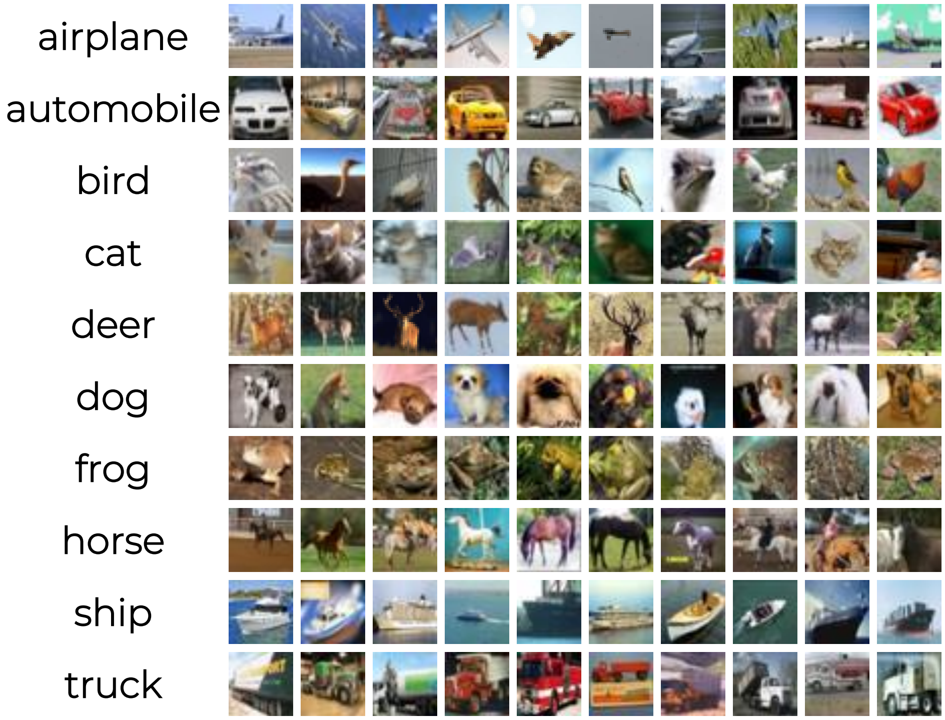
Dataset Information

Dataset: CIFAR-10

Task: Classification

Background: https://www.cs.toronto.edu/~kriz/cifar.html
The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images.

The dataset is divided into five training batches and one test batch, each with 10,000 images. The test batch contains exactly 1,000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5,000 images from each class. Here are the classes in the dataset, as well as 10 random images from each:



Model: ResNet-20 (v2), with 2x2x2 block groups of 16,64,128 filters respectively. Kernel regularization on dense and conv layers with l2(0.005) and He normal initialization. Input features are scaled to (0,1), and then each channel is centered around the dataset mean and scaled by the standard deviation.

Masterful Training Policy

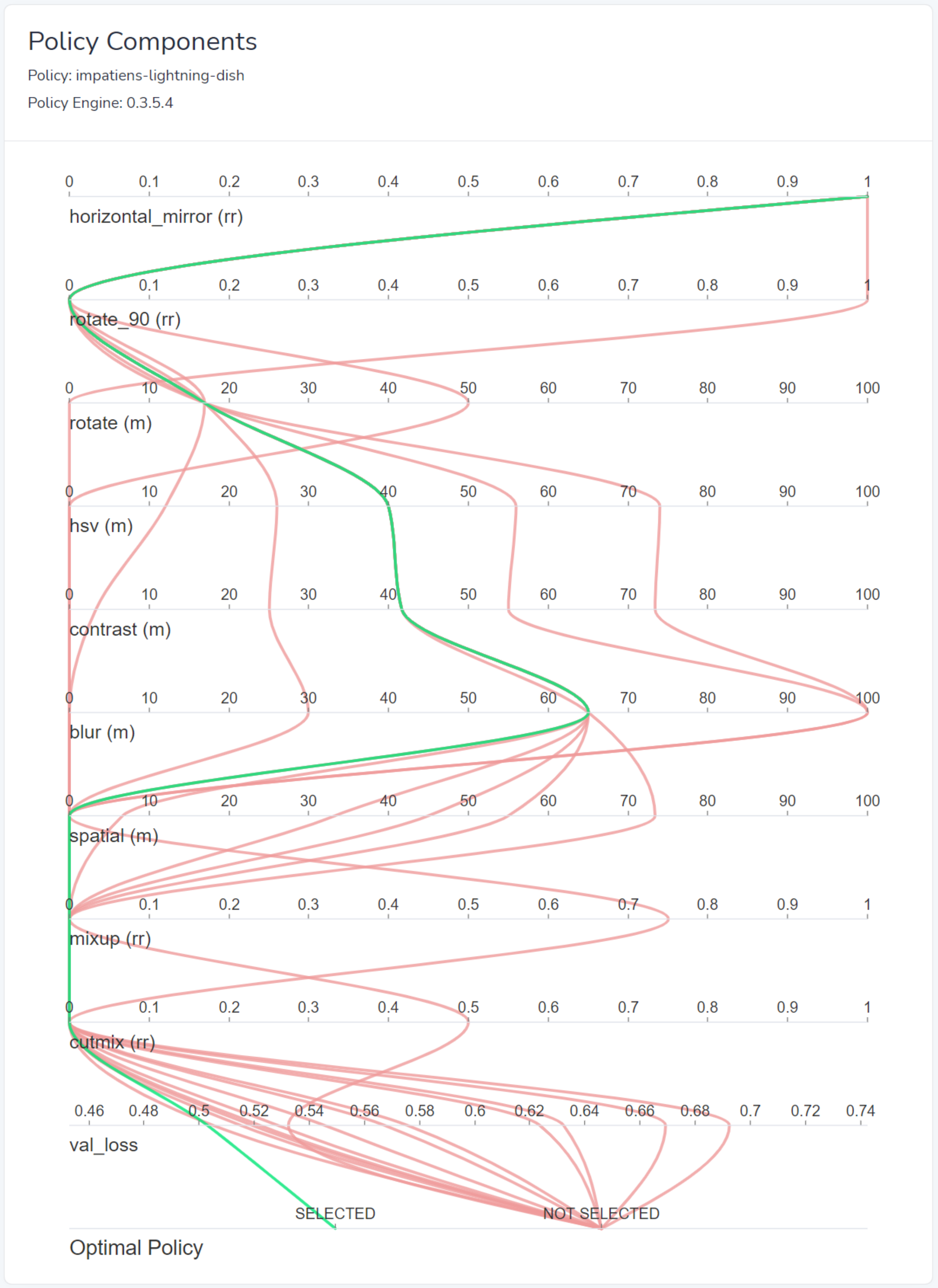
Masterful provides a parallel coordinates plot that describes the “analyze” phase of the Masterful AutoML Platform. During this phase, Masterful tests a variety of training policies on the target dataset and model using an efficient search process. While there is only one optimal policy produced, many other policy variations are examined and analyzed in the process, and the policy that yielded the best performance metrics is selected by the algorithm.

Policies that were search are shown as colored lines in the parallel coordinates plot. The green line shows the optimal policy that was found, while the red lines show policies that were tested but not selected due to lower performance. The policy name is randomly generated for each analysis run, and the policy engine is the version of the Masterful platform which generated the policy.

In the plot, each horizontal axis represents a single technique, or a compound set of techniques, that are included in the policy. The order of the horizontal axes doesn’t mirror the architecture of the model, but rather the order they were searched on by the metalearning algorithm.

There are 3 categories of values in the parallel coordinates plot. The Category is implied via a suffix to the horizontal axis legend:

1. Magnitude “(m)”: Each value on the axis shows the magnitude of the technique.
2. Replacement Rate “(rr)”: Each value on the axis shows the replacement rate of the technique. A replacement rate specifies the percentage of the dataset the technique is applied on. i.e. 0.5 means the technique was applied on 50% of the examples in the dataset, and the other 50% were not impacted by this technique.
3. Ratio “(r)”: Each value on the axis represents the ratio used for the technique. This is contextual. For example, a synthetic data blending with a ratio of 0.1 means the amount of synthetic data blended into the dataset was equal to 10% of the original dataset size. A dataset with cardinality of 100k examples, would have 10k synthetic data examples blended, and the total dataset cardinality, including original and added synthetic data, is 110k.



Policy Components

Policy: impatiens-lightning-dish

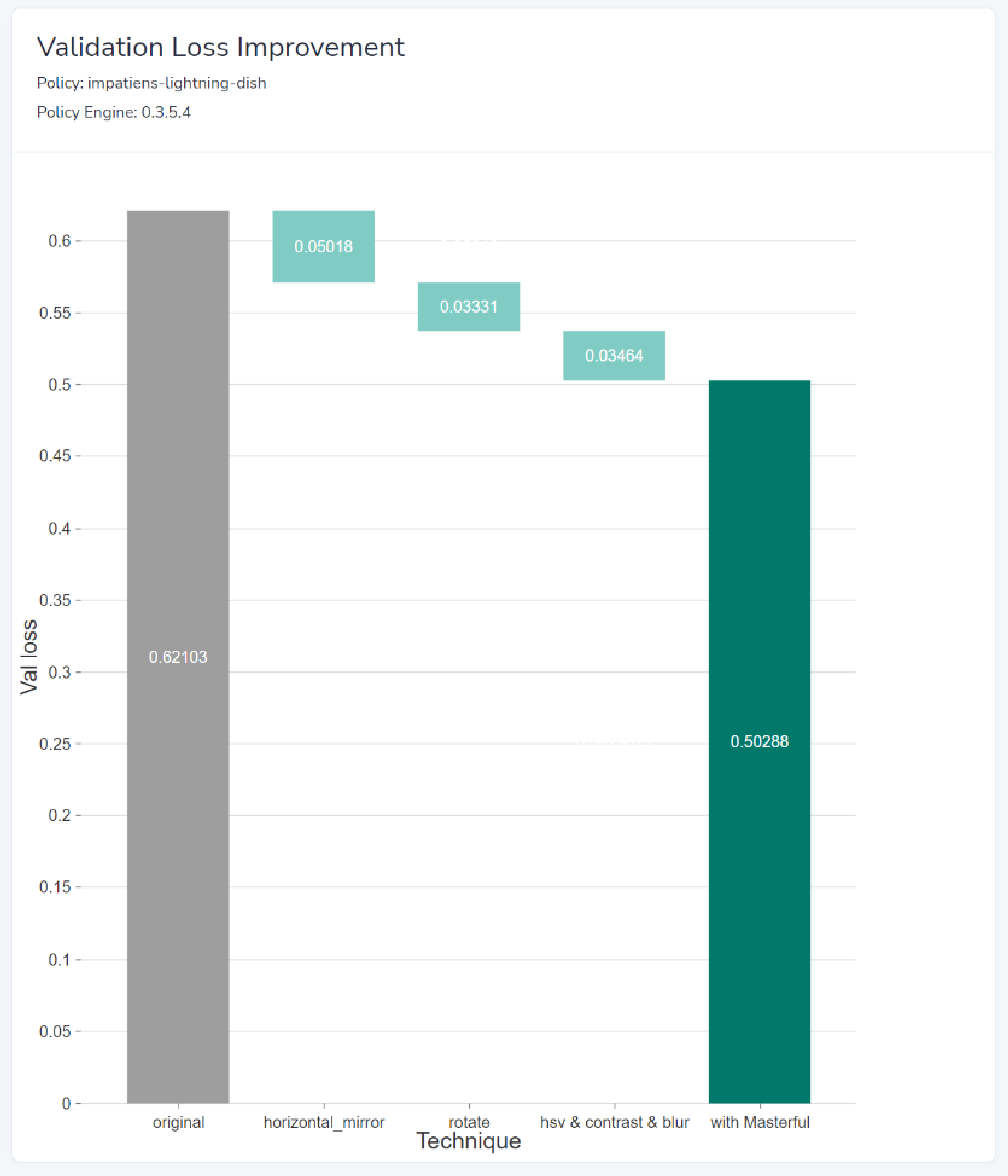
Policy Engine: 0.3.5.4

Predicted Improvements

The following charts compare two models. Both models use the same specified architecture and are trained with the same dataset. The “original” model is trained without Masterful, while the “masterful” model is trained with the Masterful platform including applying the optimal training policy.

Validation Loss Improvement

This chart plots the impact of each technique included in the optimal policy on validation loss. By visualizing the impact of techniques individually on validation loss, developers can understand which techniques applied by Masterful most increased their model performance. This gives insight into the quality of the dataset as well as the selection of model architecture.



Validation Loss Improvement

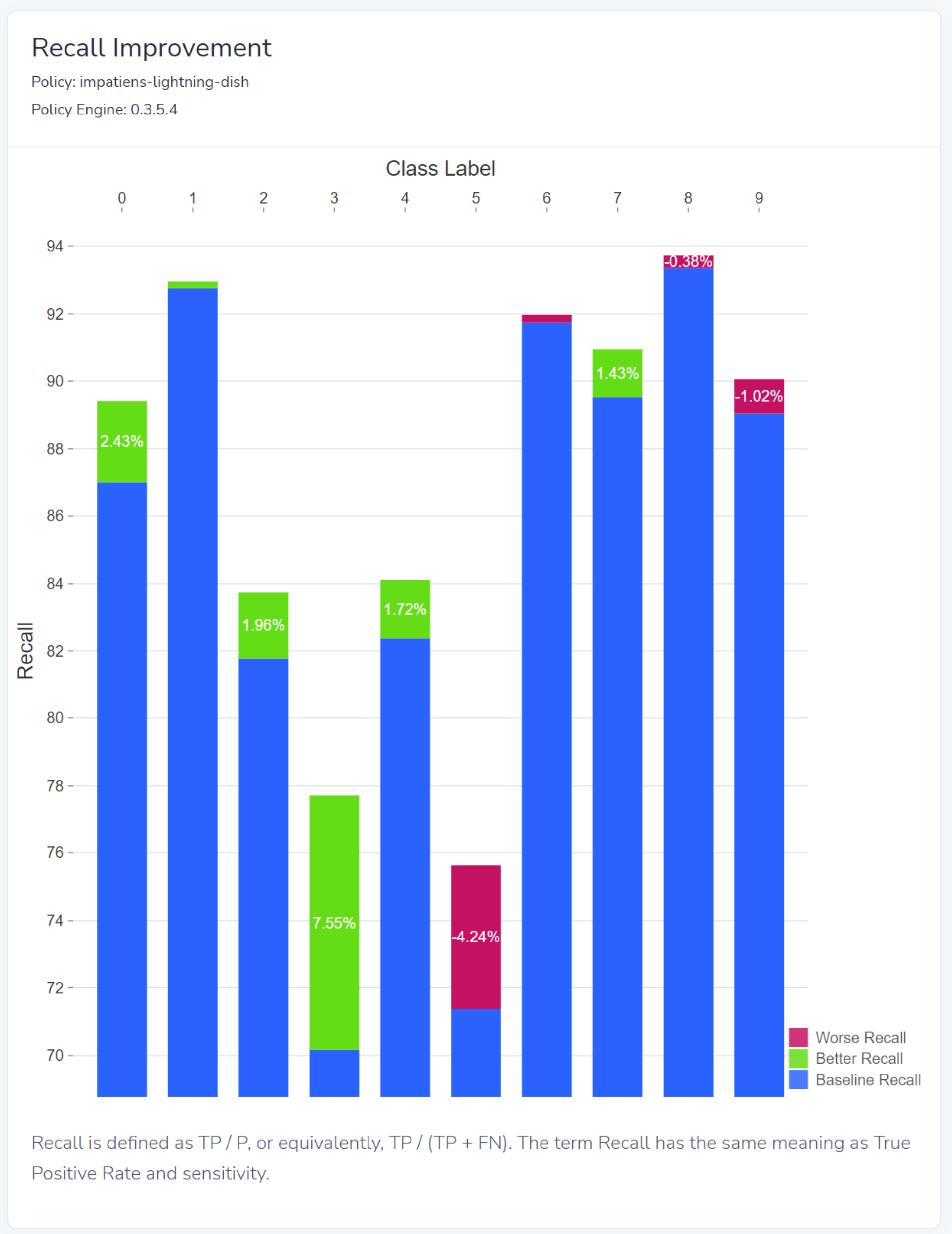
Policy: impatiens-lightning-dish

Policy Engine: 0.3.5.4

Recall and Precision Improvement

These chart shows recall and precision for each class, and Masterful's impact on each class relative to the baseline model not trained with Masterful. By understanding model performance on a per-class basis, developers are better equipped to address specific weak points of their model accuracy in addition to applying Masterful.

Consider for example a classifier model with good average performance across a 10 labels use case. The model could be very good at classifying 8 out of the 10 labels, but bad at classifying the remaining 2 labels. By looking at metrics the average all classes, developers learn nothing about the model's performance per label, and might not even be aware of the problem. This could lead them, in the pursuit of improving accuracy, to trying solutions that may not actually address the underlying performance problem, or even regress the performance of their model on other classes.

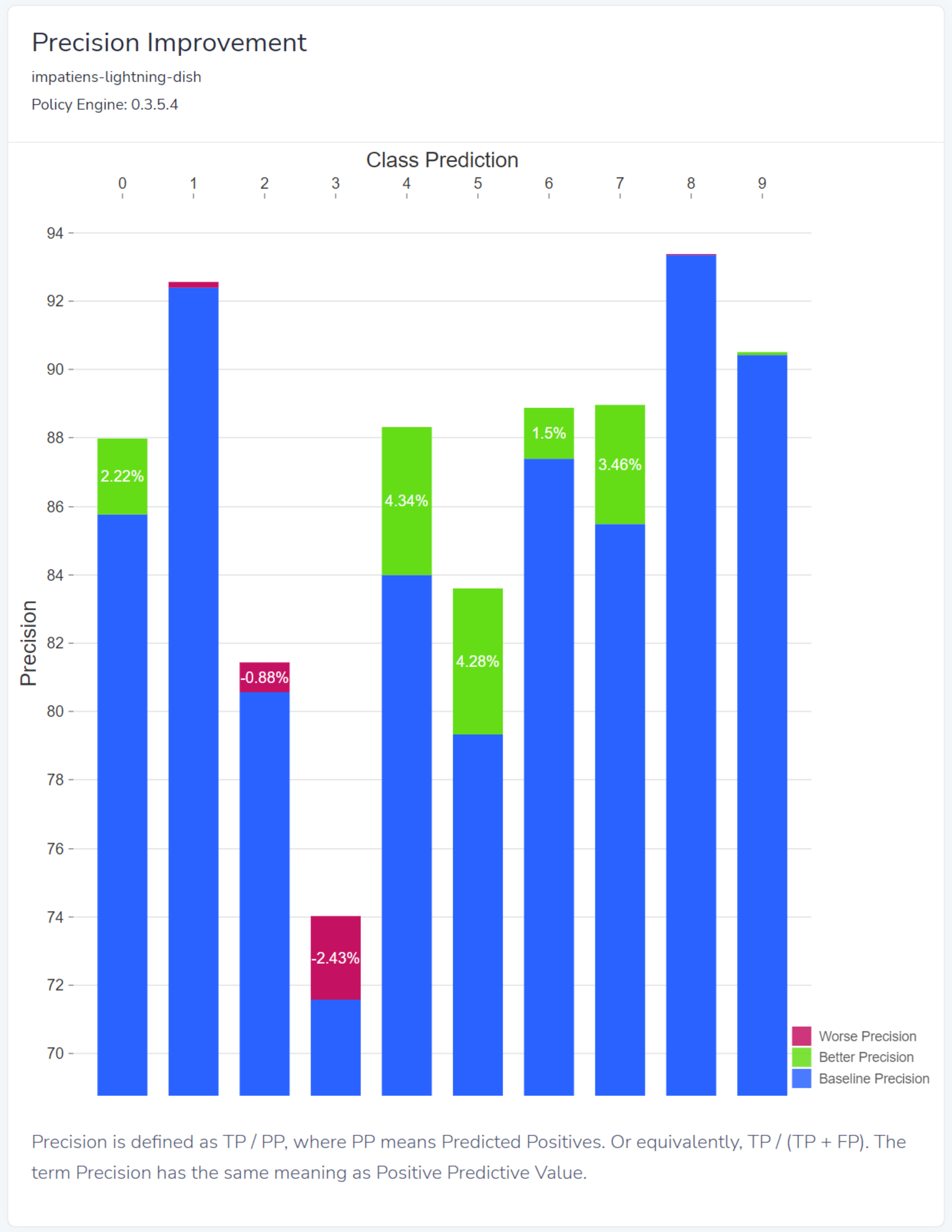


Recall Improvement

Policy: impatiens-lightning-dish

Policy Engine: 0.3.5.4

Recall is defined as TP / P, or equivalently, TP / (TP + FN). The term Recall has the same meaning as True Positive Rate and sensitivity.



Precision Improvement

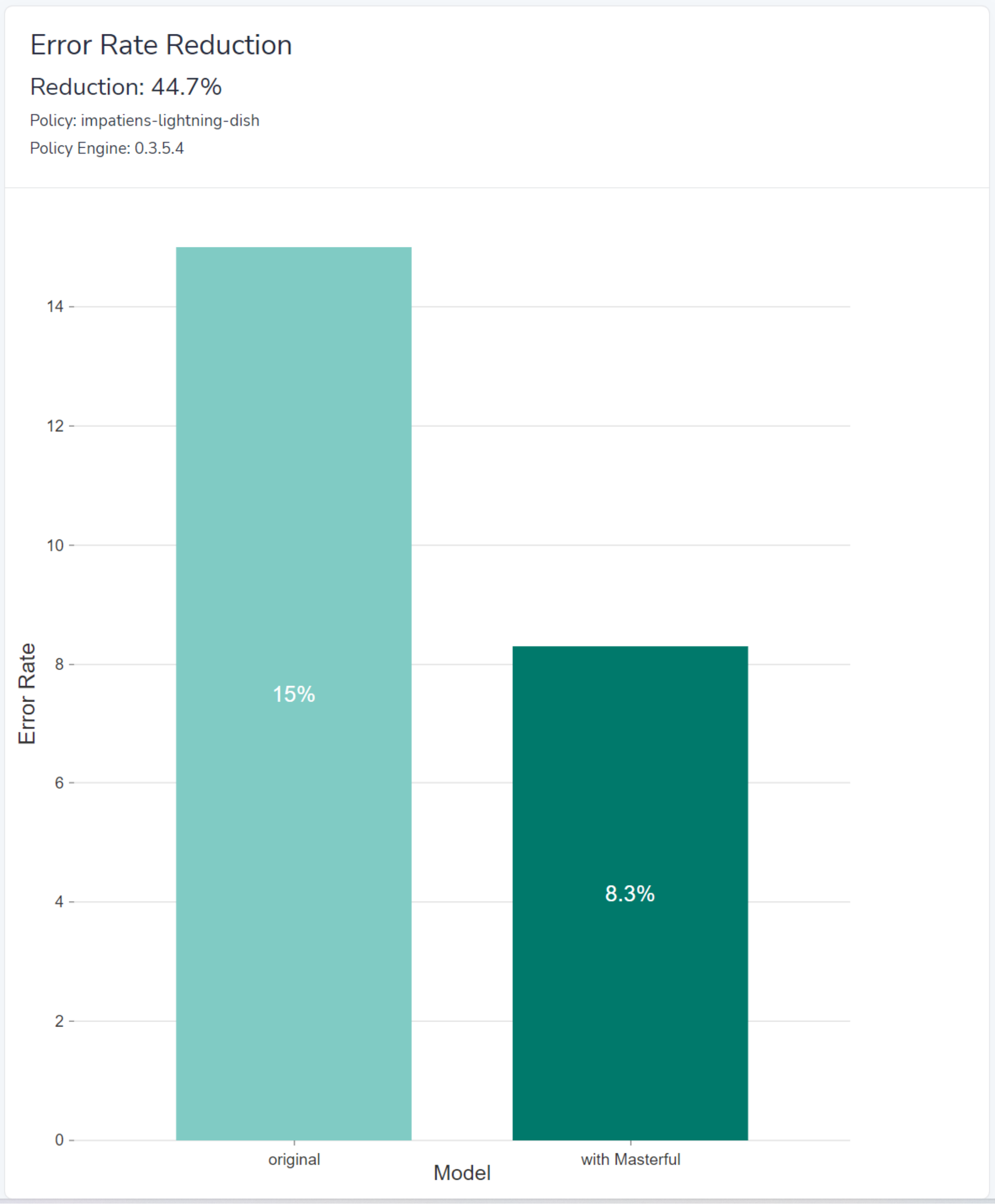
impatiens-lightning-dish

Policy Engine: 0.3.5.4

Precision is defined as TP / PP, where PP means Predicted Positives. Or equivalently, TP / (TP + FP). The term Precision has the same meaning as Positive Predictive Value.

Error Rate Reduction

This chart compares the error rate in the original model to the error rate in the model trained with Masterful. Error is defined as (1 - accuracy rate). Reduction in error rate is a more informative metric than increasing the accuracy rate. Take for example a model with a relatively high starting accuracy rate of 90% (which is an error rate of 10%). Cutting the error rate by 50%, or equivalently boosting accuracy to 95%, may be a very meaningful improvement to the model performance, especially in the case of classifying sensitive information such as the presence of a tumor.



Error Rate Reduction

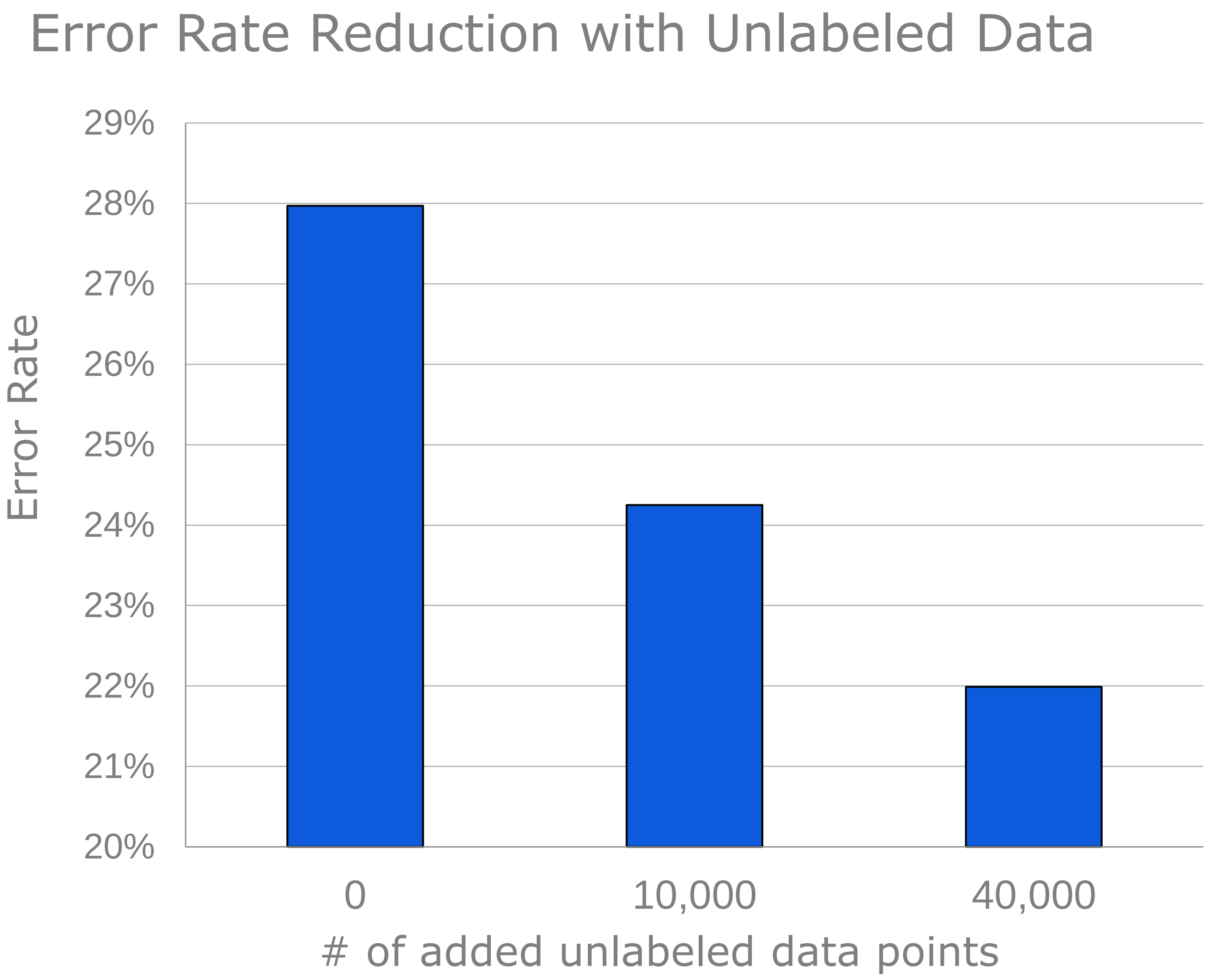
Reduction: 44.7%

Policy: impatiens-lightning-dish

Policy Engine: 0.3.5.4

Error Rate Reduction with Unlabeled Data

This chart shows the impact of training a model with unlabeled data to improve performance. The baseline model is trained with labeled data points, and then additional unlabeled data points are used to further improve the model.



Error Rate Reduction with Unlabeled Data

Improvement from training with Masterful Semi-Supervised Learning using unlabeled data drawn from the original training set. The baseline model was trained with 10,000 labeled data points.

Get Started using Masterful

Ready to get started and apply Masterful directly to your models? Go to masterfulai.com/get-started to request beta access. Masterful is installable as a Python pip package to use in your environment.