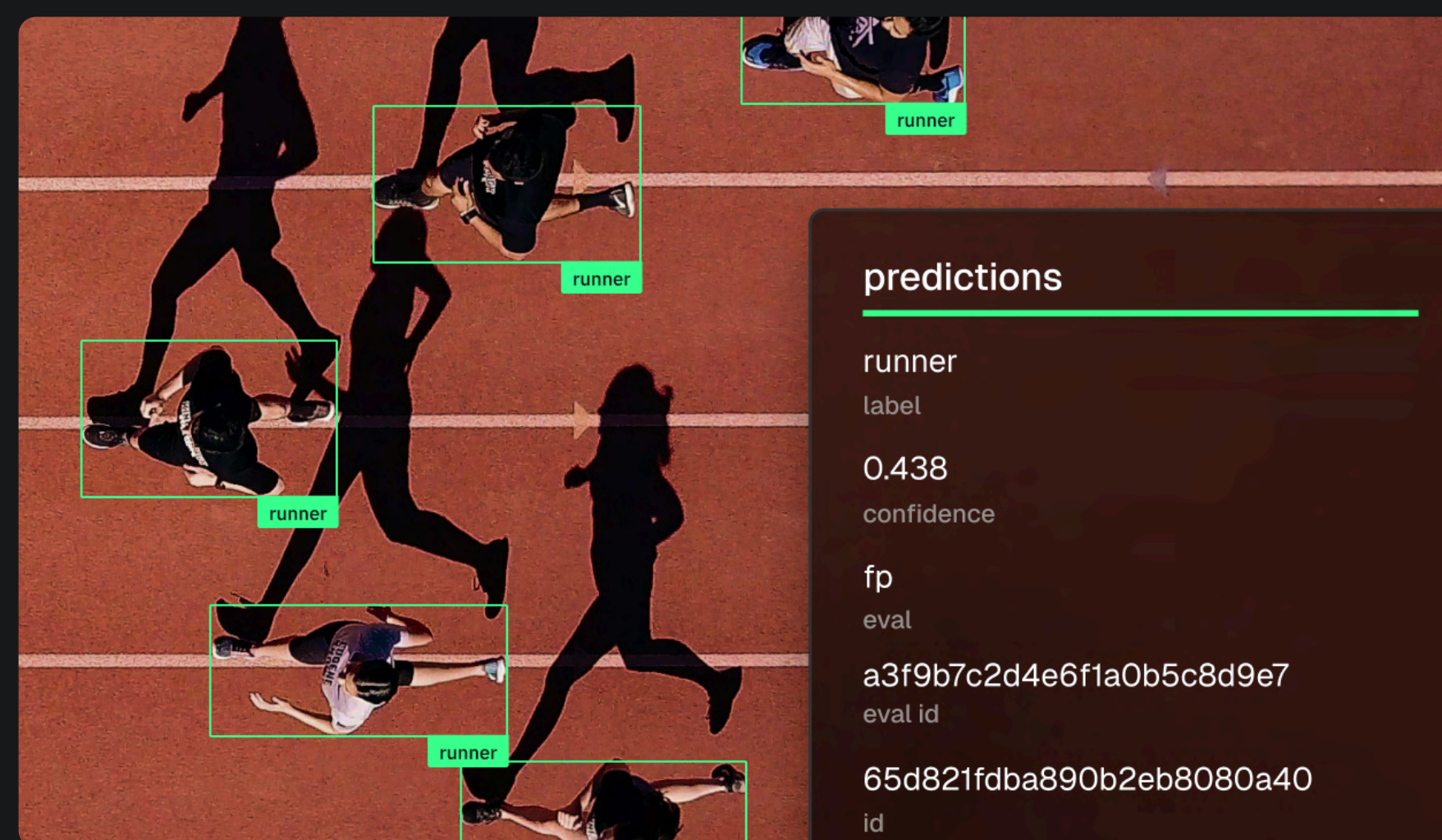
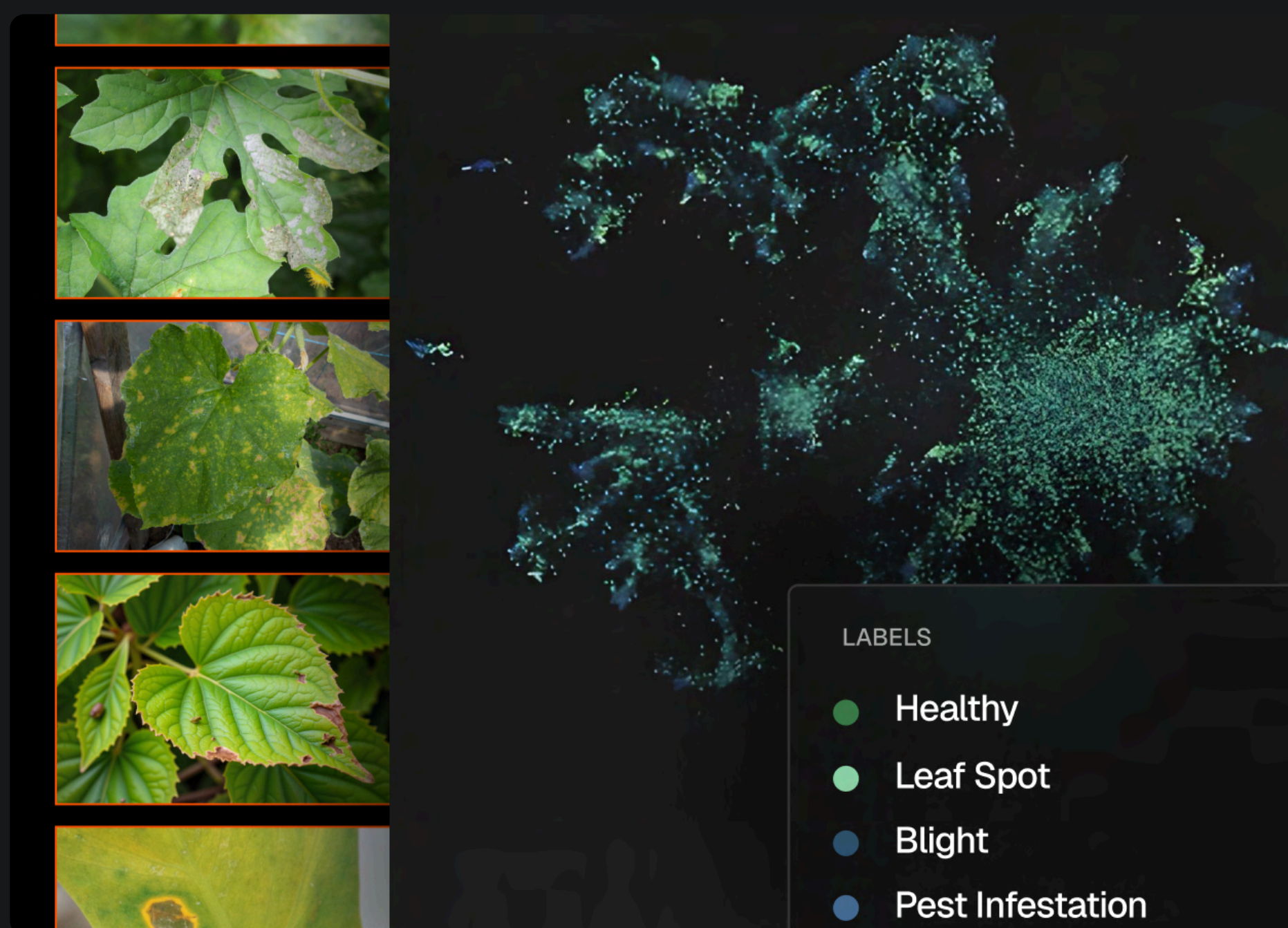


Why Vision AI Models Fail

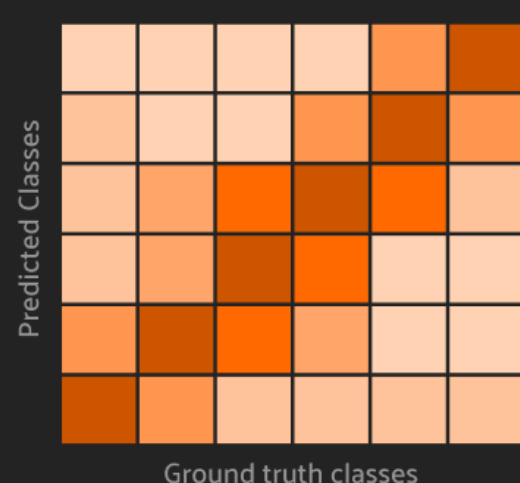
A guide to understanding common failures and how to avoid them



Download the whitepaper



Confusion Matrices



Model Performance

Indoor

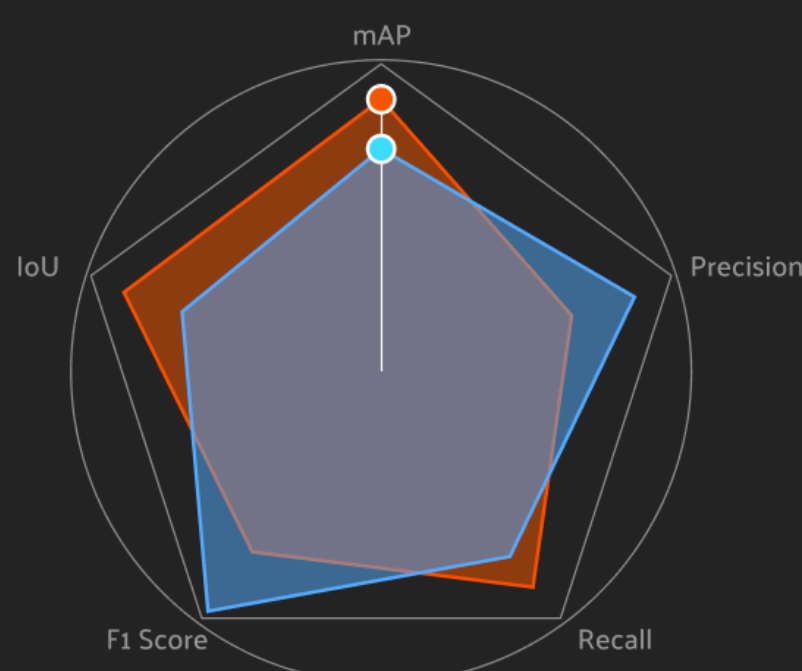




Table of contents

03	Executive summary
04	Introduction: Data > Models
04	Common model failure modes in computer vision
05	Your model is only as good as the data it learns from
06	Key data failure modes: Causes and mitigation strategies
11	Avoiding data leakage
12	Key strategies for analyzing model failures
15	Conclusion



Executive summary

A single misidentified pedestrian costs lives. A failed retail checkout system results in millions of lost revenue. When computer vision systems fail, the consequences are far-reaching.

As computer vision (CV) systems become increasingly integrated into critical business functions, the reliability of AI models is under growing scrutiny. From autonomous navigation to healthcare diagnostics, manufacturing defect detection, retail, and agriculture tech, the cost of failure is high, and downstream implications pose business risks.

Several prominent incidents have been traced to AI model inadequacies:

- Tesla's Full Self-Driving (FSD) system struggled to detect pedestrians and obstacles in low-visibility conditions, leading to several crashes.
- Walmart's AI system for anti-theft prevention reported incidents of falsely flagged innocent customer behaviors as theft, while failing to detect actual shoplifting scenarios.

- Taiwan Semiconductor Manufacturing Company's (TSMC) initial approaches for automated wafer inspection did not perform well on certain defects, revealing gaps in the defect pattern coverage that affect wafer yields and downstream costs.

The harsh reality:

Even the most advanced models often fail due to issues linked to poor data quality, underrepresented edge cases, model bias, and many other factors.

Building more robust, trustworthy AI systems requires more than just architectural improvements. A strong foundation in data curation, model evaluation, and analysis is key to preventing failures. This guide outlines the most common failure modes in modern vision systems, unpacks the real-world consequences they can cause, and shares proven strategies that leading AI teams use to detect, debug, and prevent model failures before they hit production.

What you'll learn

By the end of this guide, you'll understand:

- The five most common model failure modes that jeopardize production vision systems.
- Real-world case studies showing how failures translate to business losses.
- Prevention strategies and techniques used by leading organizations.
- An evaluation framework for bulletproofing your deployments.



Introduction

Computer vision has advanced rapidly thanks to large-scale vision models, multimodal architectures, diverse datasets, as well as the underlying compute infrastructure. But even as top-line accuracy metrics improve, many vision models are susceptible to failures. Some estimates quote that more than 80% of AI projects fail.

Failures can stem from numerous reasons. Prominent ones include poor data labeling, lack of diversity in training sets, distribution shifts, or fragile architectures. These issues not only deteriorate model accuracy but also create significant downstream operational disruptions and business risks.

Understanding where and why vision models fail is critical for building resilient, production-grade systems. The patterns are predictable and preventable.

Common model failure modes in computer vision

Despite high benchmark performance, numerous model failure modes have been observed in real-world deployments. Incidents in the medical field, such as the Epic Sepsis Model (ESM) used for predicting sepsis onset, missed almost two-thirds of actual cases and generated false alarms due to sample selection bias.

These incidents demonstrate the dangers of deploying models that have not gone through extensive external validation.

Model failures can be carefully traced to mistakes at different stages of the machine learning (ML) lifecycle.

SafelyYou improves elderly patient safety with data-centric visual AI

SafelyYou AI-powered solutions utilize computer vision models and data pipelines to process millions of videos and analyze over 350,000 real-world events. Using FiftyOne's data analysis capabilities to root cause

AI model failures, Safely You resulted in an 80% reduction in fall-related ER visits while driving patient safety and cost savings. [Learn how they achieved this.](#)



The four critical failure categories

1. Data-related failures:

Errors include mislabeled data, poor quality or biased datasets, lack of coverage for real scenarios, and data leakage between splits. These types of failures lead to models learning incorrect or overly narrow patterns.

2. Model training failures:

Overfitting or underfitting to training data, class imbalances, or the use of incorrect models/parameters can all cause real-world failure.

3. Evaluation failures:

Overlooking data leakage, lack of testing on a diverse set of scenarios, or a focus on aggregate metrics alone can mask serious flaws. Models might look accurate on certain benchmarks and parameters, but might be biased or unstable in specific scenarios.

4. Deployment failures:

Data drift, lack of monitoring, or not keeping models updated are common pitfalls of model failures.

Each category represents a different stage where things can go wrong, but data-related failures are often the most dangerous because they're the hardest to detect.

Your model is only as good as the data it learns from

Unlike model architecture bugs or training instability, which tend to surface early, failures rooted in data can silently degrade performance in critical situations. Common examples include biased sampling, label noise, or poor coverage of edge cases.

- **Why data failures are so dangerous:**

These failures can propagate downstream, leading to high-confidence but incorrect predictions, especially in safety-critical applications like autonomous driving, healthcare, or industrial automation.

Worse, because the model appears to be functioning correctly on standard benchmarks, teams may not realize the extent of the issue until it causes real-world harm. Investing in rigorous data curation, validation, and continuous monitoring is essential to mitigating risks. After all, models are only as good as the data they learn from.



Key data failure modes: Causes and mitigation strategies

1. Insufficient or unrepresentative data

The problem

A model trained on a dataset that's too small or one that lacks coverage may perform well on training and test data, but fail to generalize. Such models become “accurate but ineffective” as they overfit to a narrow set of samples but often miss real-world variation. For instance, an object detector trained on only a few hundred images might appear to learn, but then miss most objects in novel scenes.

Real-world scenario

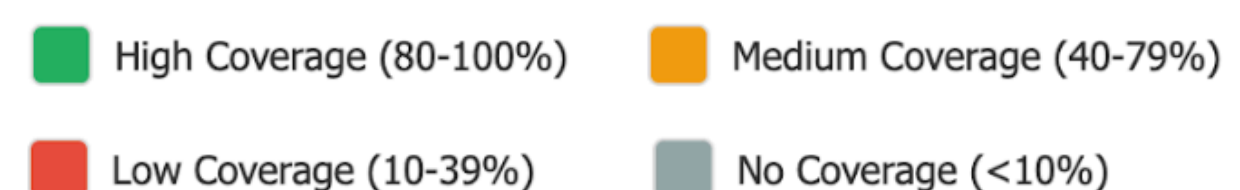
Reports from November 2023 revealed GM's Cruise robotaxi couldn't consistently recognize children or large holes, causing a major pedestrian accident. Its model training data lacked representation of these edge cases in its training set.

Best practice

Gather a sufficiently large and diverse dataset that covers a range of conditions the model could face, e.g., different lighting conditions, weather, backgrounds, etc. For niche use cases, dataset augmentation using synthetic data can achieve broader scenario coverage.

Hot tip

Use powerful libraries and tools such as [Albumentations](#), [NVIDIA Omniverse](#) for data augmentation and synthetic data generation. Perform deeper coverage analysis by pairing them with [FiftyOne curation](#) and [model evaluation](#) capabilities, which include tagging, filtering, embeddings visualization, and dataset statistics to visually inspect scenario coverage, e.g., how many samples exist per lighting condition, weather tag, or object size.



Training Dataset Coverage					
	Daylight	Dusk/Dawn	Night	Rain/Snow	Edge Cases
Adult Pedestrians	95%	65%	45%	30%	40%
Children	60%	25%	8% ⚠	5% ⚠	2% ⚠
Vehicles	90%	80%	70%	55%	45%
Road Obstacles	70%	50%	35%	25%	20%
Large Holes/Potholes	25%	15%	8% ⚠	3% ⚠	1% ⚠

2. Imbalanced class distribution

The problem

This is a common failure mode where the model performs well on aggregate accuracy (dominated by the well-represented class) but has blind spots for the under-represented cases. Take an example of a driving dataset that has thousands of “car” images but only a few “bicycle” or “bus” images. A model trained on this data will be great at the frequent classes and not so great at the rare ones.

Real-world scenario

Amazon’s Just Walk Out system was discontinued in U.S. stores in 2024 after [frequent checkout errors](#), particularly when customers returned items to different shelves, shopped in groups, or picked up rare products. These failures stemmed from class imbalance in the vision model, which was heavily trained on common item interactions but lacked sufficient examples of these less frequent behaviors, causing the system to misclassify actions and trigger manual audits.

Best practice

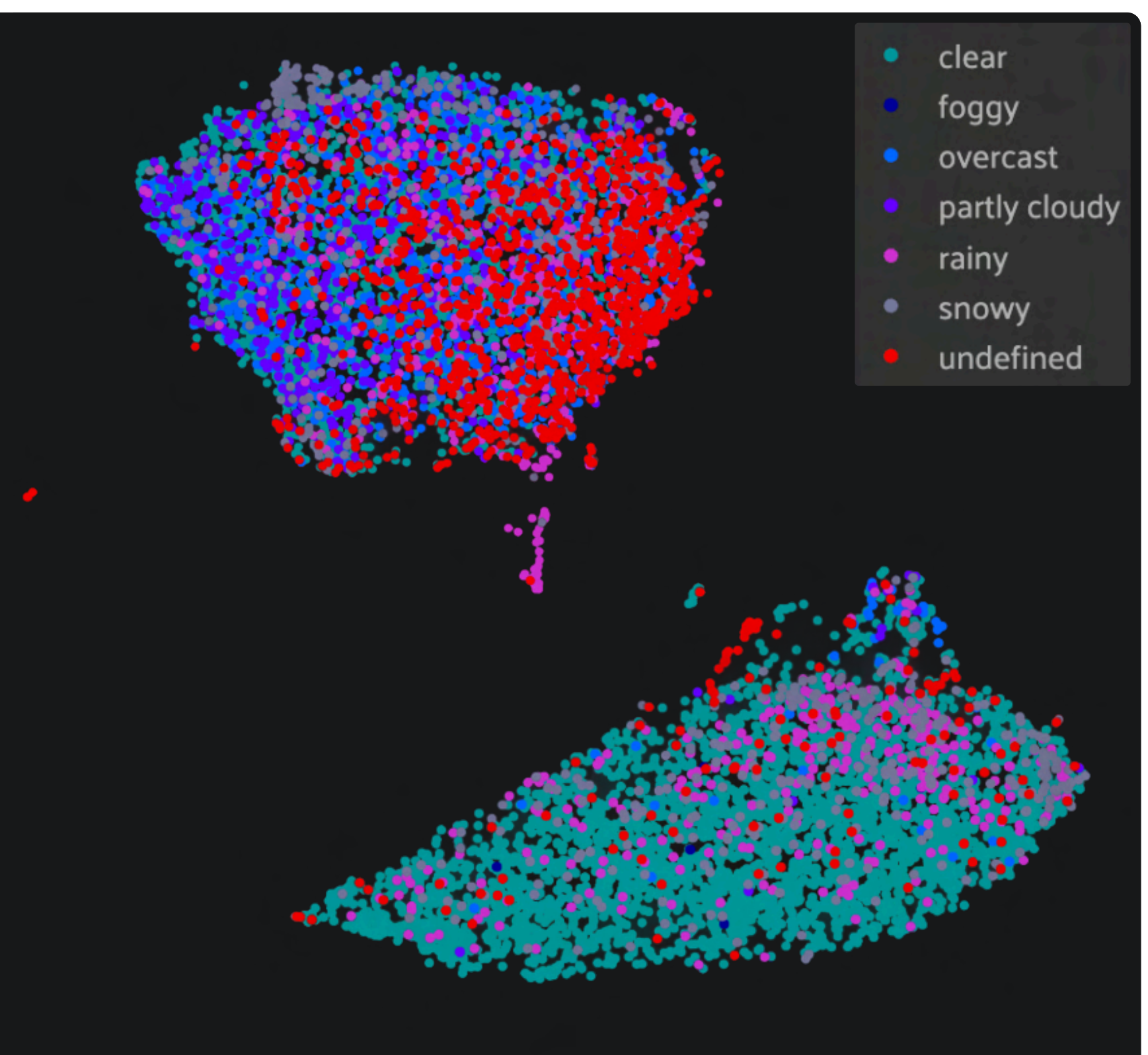
Balance the dataset where possible. This can mean collecting more samples of rare classes or down-sampling the dominant class. Data augmentation or synthetic data generation for minority classes can also help. Additionally, use class-aware training strategies (like loss weighting) so that the model pays attention to all classes.

- **Hidden Danger:**
Some imbalances are concealed. For example, if all images of cars in your dataset are taken during the day and from the front, the model may struggle to recognize cars at night or from other angles. Even if the class count looks balanced, a lack of variety in context can skew the model's understanding. Careful curation is essential to ensure each class appears in diverse settings.

Hot tip

Use tools like Scikit-learn’s [compute_class_weight](#) or PyTorch’s [weighted loss functions](#) to account for class imbalance during training. These help ensure the model doesn't default to predicting the majority class just to boost overall accuracy.

For deeper insight into hidden or contextual imbalances, the [FiftyOne Embeddings workflow](#) helps visualize clusters of similar samples across datasets. This lets you uncover whether certain classes only appear under specific conditions (e.g., angle, lighting), even when class counts appear balanced.





3. Noisy or low-quality samples

The problem

Blurry, duplicates, or poorly exposed images can make it harder for computer vision models to extract essential information and mislead the training process. A model trained on suboptimal images may misclassify objects when subjected to clear images since it learned “fuzzy” patterns.

Real-world scenario

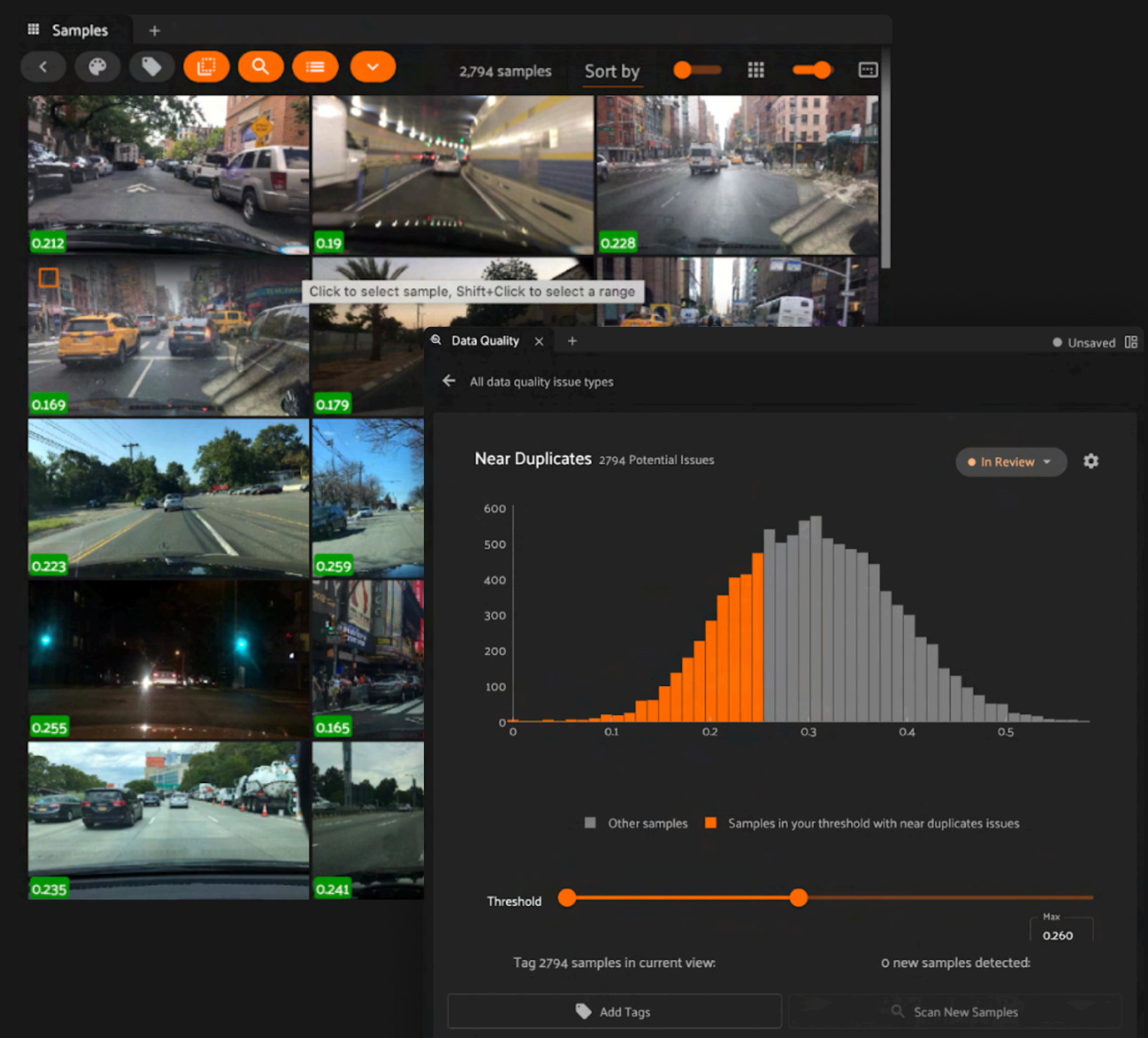
Analysis shows that the inclusion of blurry and duplicate images in the ChestX-ray14 dataset significantly compromised model training. These poor-quality inputs masked overfitting and inflated performance during validation. As a result, models failed to generalize in real-world clinical settings, misdiagnosing conditions and underperforming on images from diverse machines and patient populations.

Best practice

Set up data quality workflows to weed out poor quality samples, such as bright, blurry, duplicates, or samples that have extreme aspect ratio or entropy. Models trained on clean, relevant data are proven to be more accurate and reliable.

Hot tip

Python libraries such as [python-pillow](#) or OpenCV library `cv2.laplacian()` can [detect blurry images](#) or visually extreme images. If you want a more [out-of-the-box data quality workflow](#) that handles all aspects of data quality, tools such as FiftyOne can detect and weed out poor-quality samples for cleaner dataset preparation.



4. Data labeling errors and inconsistencies

The problem

Even a small percentage of mislabeled data can prompt the model to learn incorrectly and lead to systematic prediction errors. For example, a scene marked as nighttime instead of daytime or a loose bounding box around an object would cause an incomplete or incorrect ground truth to be recorded. A clean, well-annotated dataset is critical; otherwise, the model is learning from flawed information.

Real-world scenario

A gender shades [research study](#) exposed a critical problem in commercial facial analysis systems from IBM, Microsoft, and Face++ and found significant accuracy disparities across gender and skin tone. These systems were trained on datasets with poor demographic representation, particularly lacking darker-skinned and female faces.

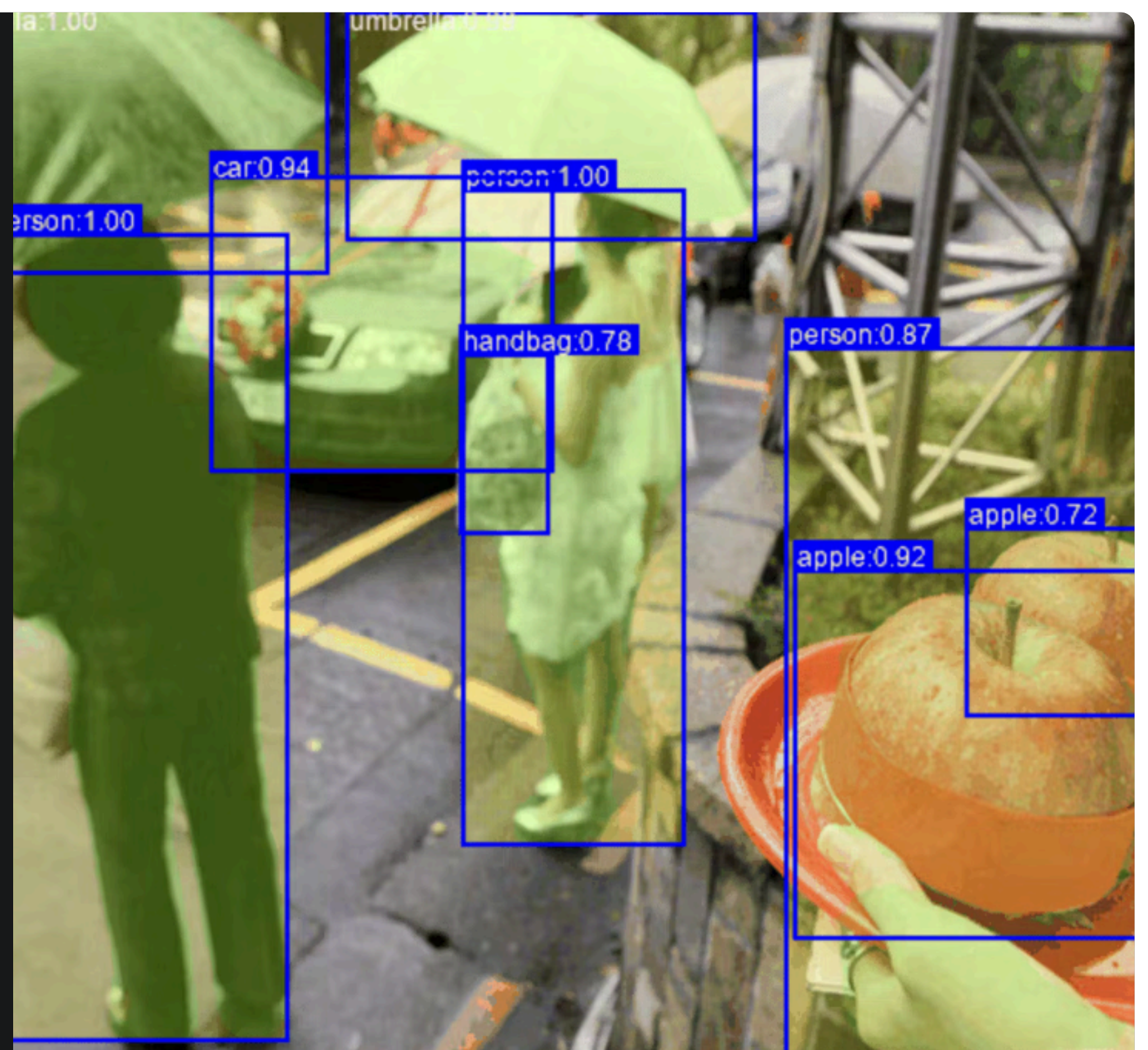
As a result, these models misclassified darker-skinned women up to 35% of the time, compared to less than 1% error for lighter-skinned men, highlighting how biased training data or incomplete labels can lead to discriminatory AI outcomes.

Best practice

Invest in high-quality labeling. If using manual annotation, provide clear guidance and use QA workflows to weed out poor-quality labels. Auto-labeling techniques are becoming increasingly popular as they provide similar or better results at a fraction of the cost. Invest in techniques that provide intelligent AI scoring and verification techniques for cleaner object detection or segmentation tasks.

Hot tip

Open-source tools like CVAT can identify labeling errors early. For faster iteration, FiftyOne's [Verified Auto-Labeling](#) combines foundation model predictions with QA filters to generate high-quality labels while automatically identifying annotation errors like incorrect classes, missing objects, or imprecise boxes at a fraction of the time and labeling costs.





5. Data bias and lack of diversity

The problem

A dataset that overrepresents certain demographics, settings, or contexts can lead to biased model behavior. Even with sufficient volume, if most examples come from the same environment or group, the model learns skewed patterns and fails in unfamiliar scenarios. For instance, a face detection model trained on only certain demographics will show bias and not perform when subjected to diverse images.

Real-world scenario

Reports from January 2025 indicate that Detroit police wrongfully arrested two men based on faulty facial recognition. The system misidentified Black individuals due to poor performance on low-quality or non-white facial images.

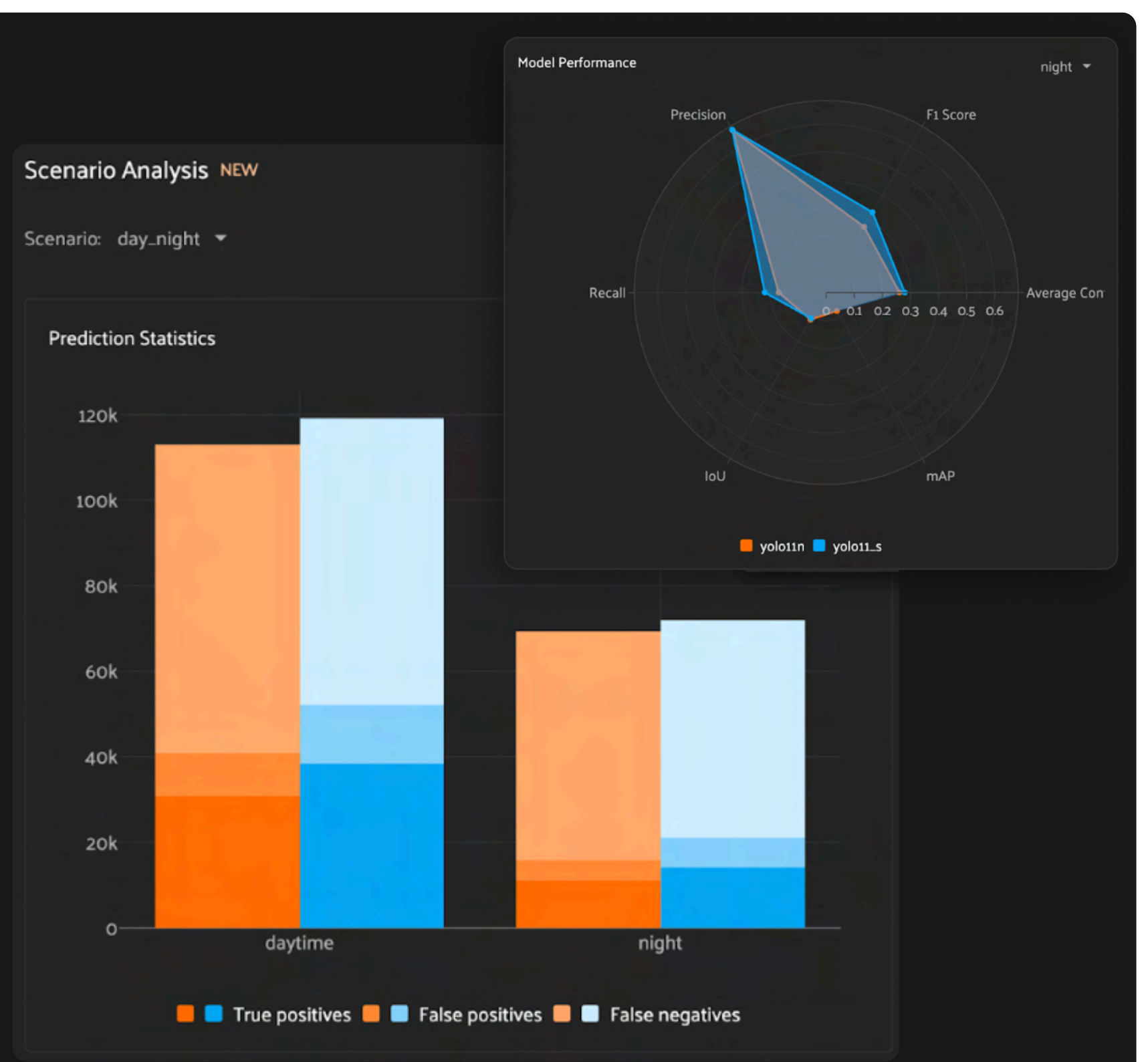
Best practice

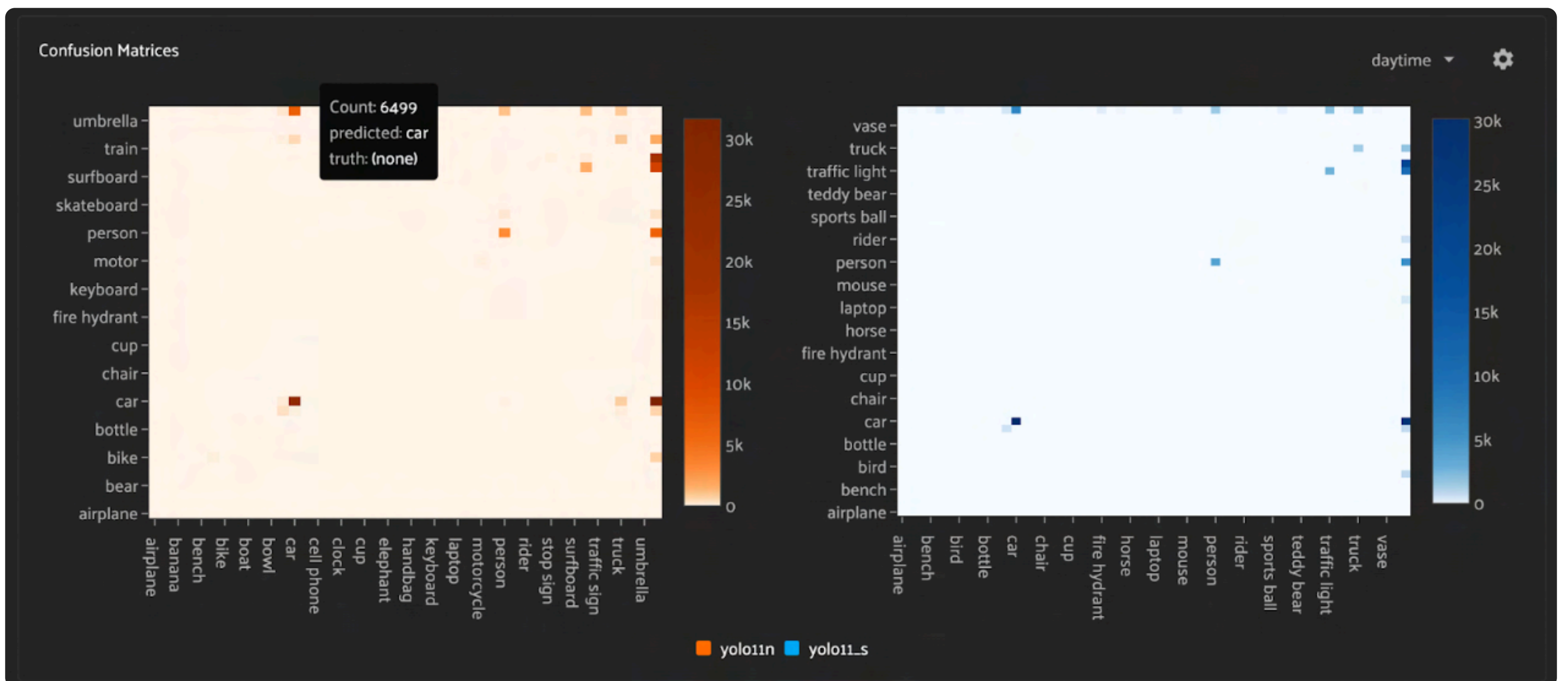
Actively seek dataset diversity during data collection. Include varied weather conditions, geographies, camera angles, and population groups so your model doesn't inherit or amplify real-world bias. Diversity helps the model learn robust, generalizable features and reduces the risk of discriminatory predictions.

Hot tip

Python libraries such as [Fairlearn](#) can audit models or datasets for demographic bias and performance disparities across subgroups. These libraries help you evaluate fairness metrics such as disparate impact or equalized odds, and can guide rebalancing strategies.

For visual datasets, combine this with [FiftyOne's curation capabilities](#), including dynamic filtering and custom tags to inspect how model performance varies across different scenes, demographics, or conditions.





Compare predicted vs actual values for the dataset across different classes. This type of visual subset analysis helps understand where the model might be getting confused across different classes.

Avoiding data leakage

The problem

A common oversight in dataset preparation is mixing data between training, validation, and test sets. If the splits are not independent, model performance metrics won't be a true indicator of real scenarios. A common best practice for splitting datasets is to allocate approximately:

- 70–80% for training
- 10–15% for validation
- 10–15% for testing

The training set is used to fit the model, the validation set helps tune hyperparameters and prevent overfitting, and the test set is reserved for final performance evaluation.

Useful strategies

- **For small datasets:**
Use k-fold cross-validation, where the data is divided into k subsets and the model is trained and validated k times, each time using a different subset as the validation set and the rest for training.
- **For imbalanced datasets:**
Use stratified sampling to ensure each split maintains the same proportion of classes as the original dataset.
- **For sequential or video data:**
Use time-ordered or scene-based splits to avoid data leakage.



Key strategies for analyzing model failures

Understanding failure modes is only half the battle. ML teams need systematic approaches to detect, analyze, and prevent them. Here are the proven strategies that separate successful CV deployments from failures.

Quantitative evaluation

What to track

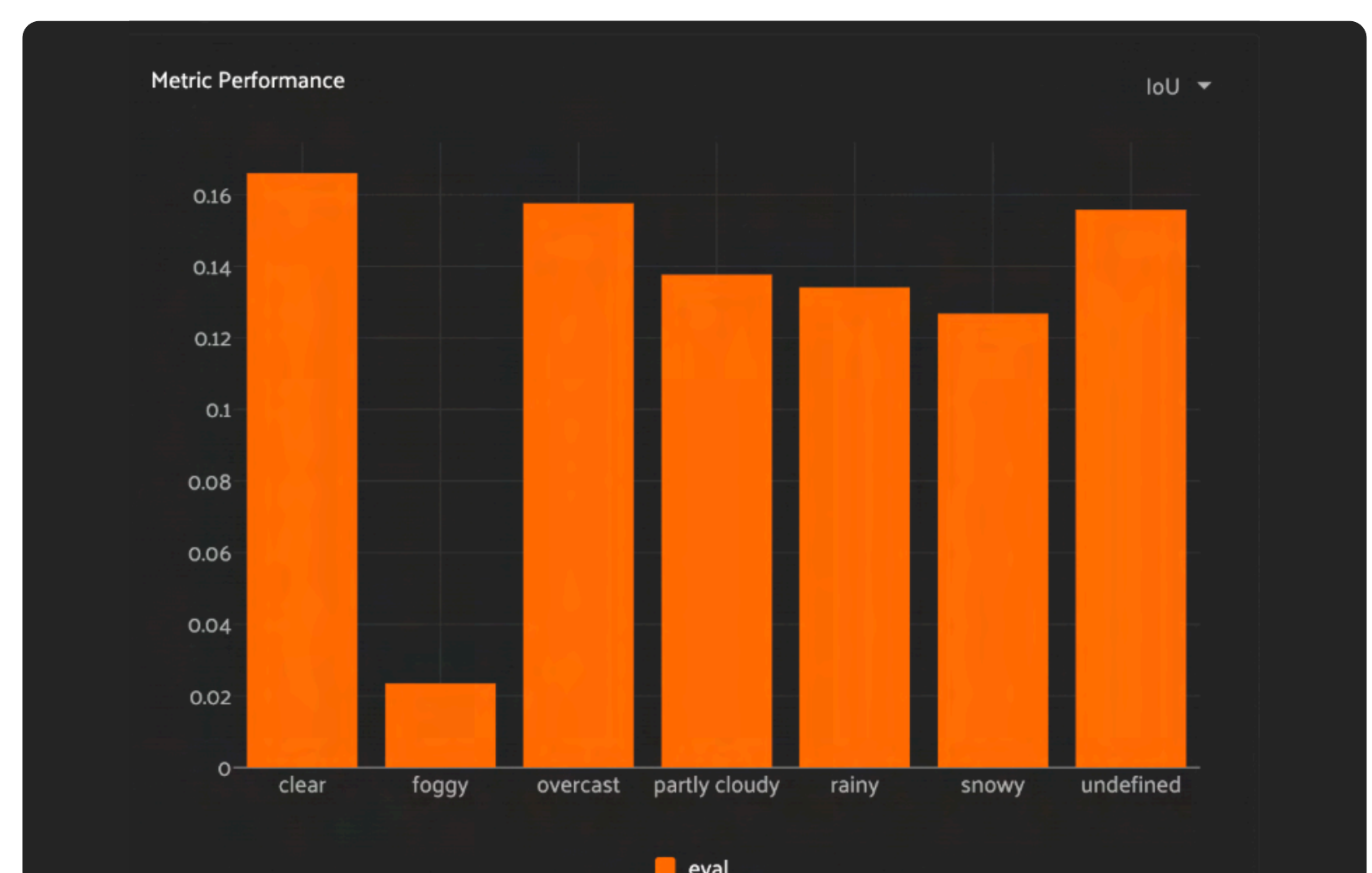
Metrics like precision, recall, F1 score, and Precision Recall (PR)/Receiver Operating Characteristics (ROC) curves identify imbalances and optimize thresholds. Use the right tools to compute and visualize these evaluation metrics side by side with the visual data.

Techniques

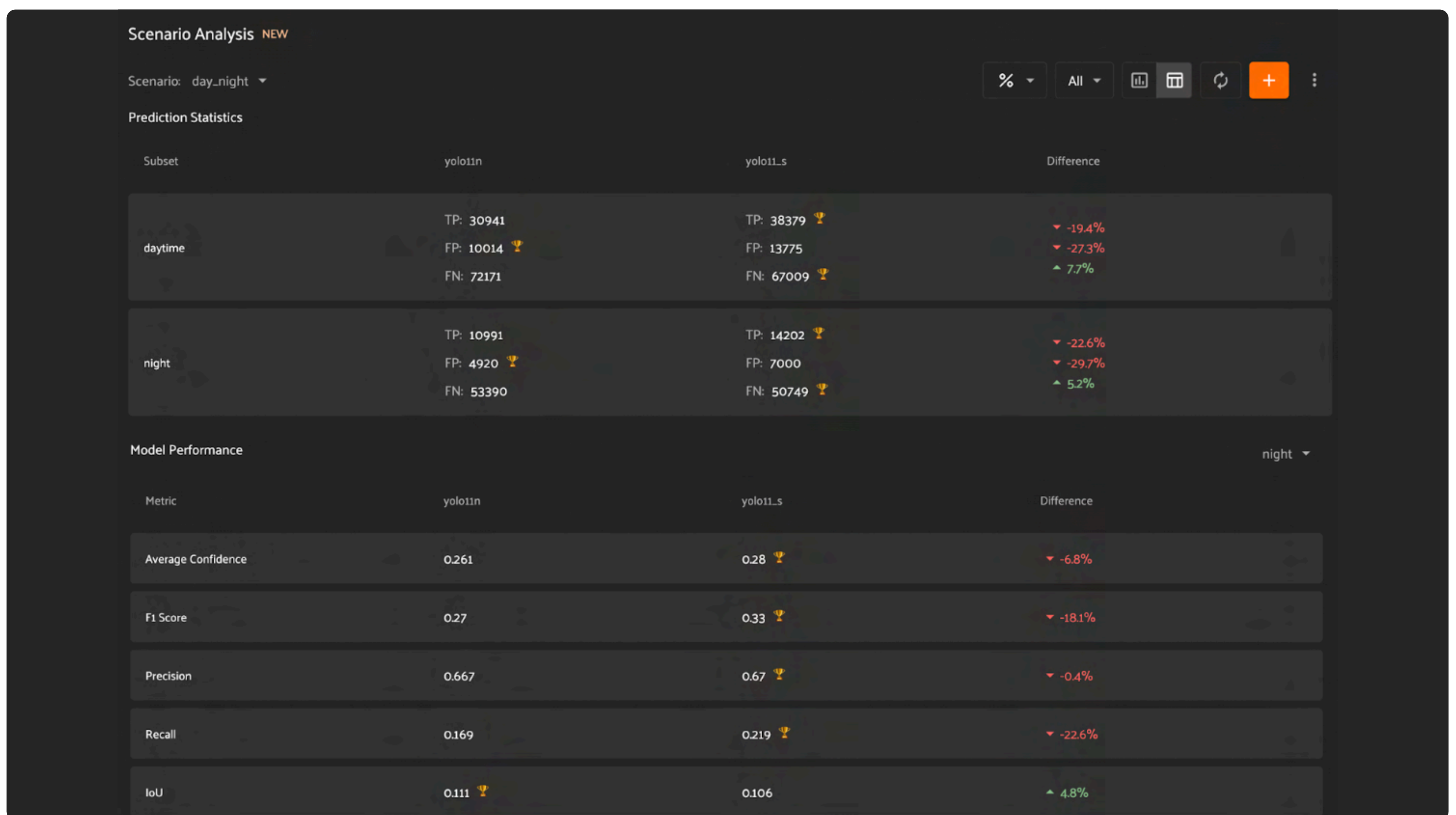
Scenario-based analysis can diagnose failures that only appear under specific contexts. Pairing these metrics with visual inspection tools is useful to ground quantitative insights in real data and prioritize what to fix.

Why It matters

These metrics help surface class imbalances, threshold sensitivity, and general performance gaps. Breaking down false positives and false negatives reveals where the model is wrong or missing critical detections. Reviewing per-class performance pinpoints which categories are underperforming.



Visually compare model performance across different weather conditions. Here, IoU metric is compared across weather scenes.



Analyze model prediction statistics across multiple models and scenes, e.g., daytime and nighttime. In the above tabular representation in the FiftyOne app, model performance metrics are compared across nighttime scenarios.

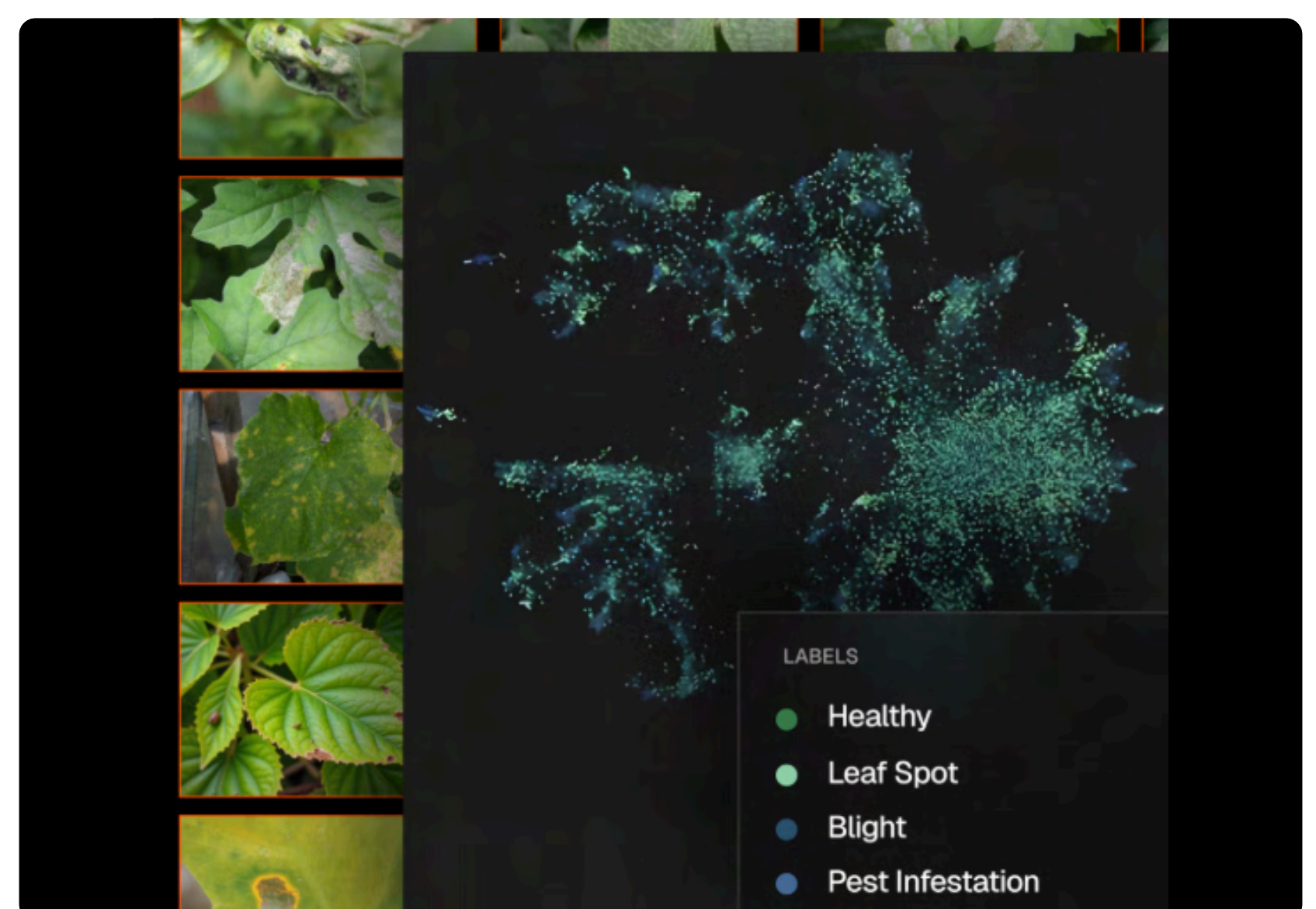
Error analysis

The power of visualization

Visualizing data in the embeddings space helps understand groupings, outliers, and mislabeled images. Embeddings project high-dimensional data into a lower-dimensional space (e.g., t-SNE or UMAP) to visually cluster samples and identify patterns.

What to look for

A good tool provides the flexibility to slice and dice the data (e.g., by confidence thresholds, classes, scenes) and view it in the context of the actual images to find errors.



Visualizing data labels in the embeddings space, side by side with the actual images from the dataset.



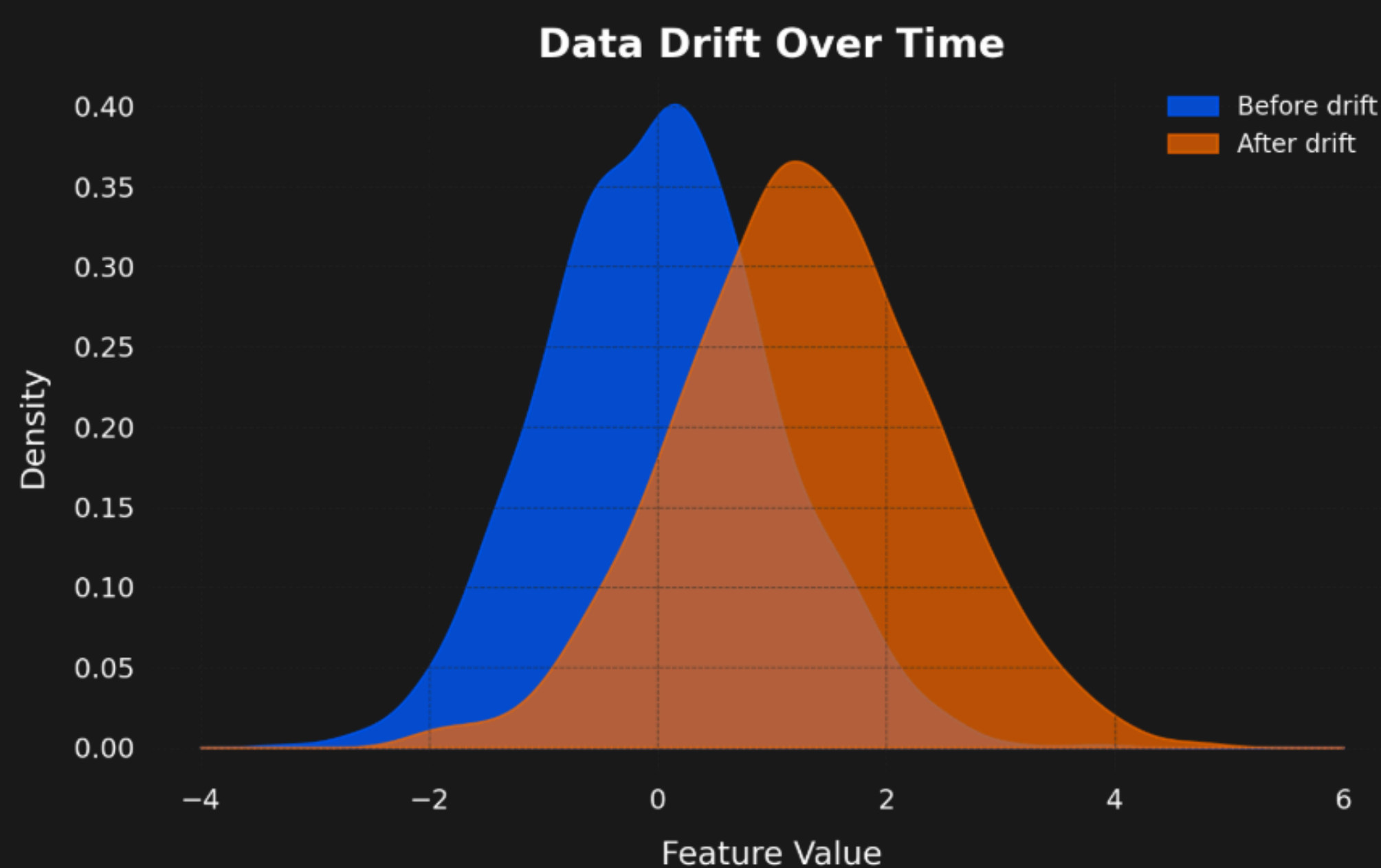
Production monitoring

Continuous vigilance

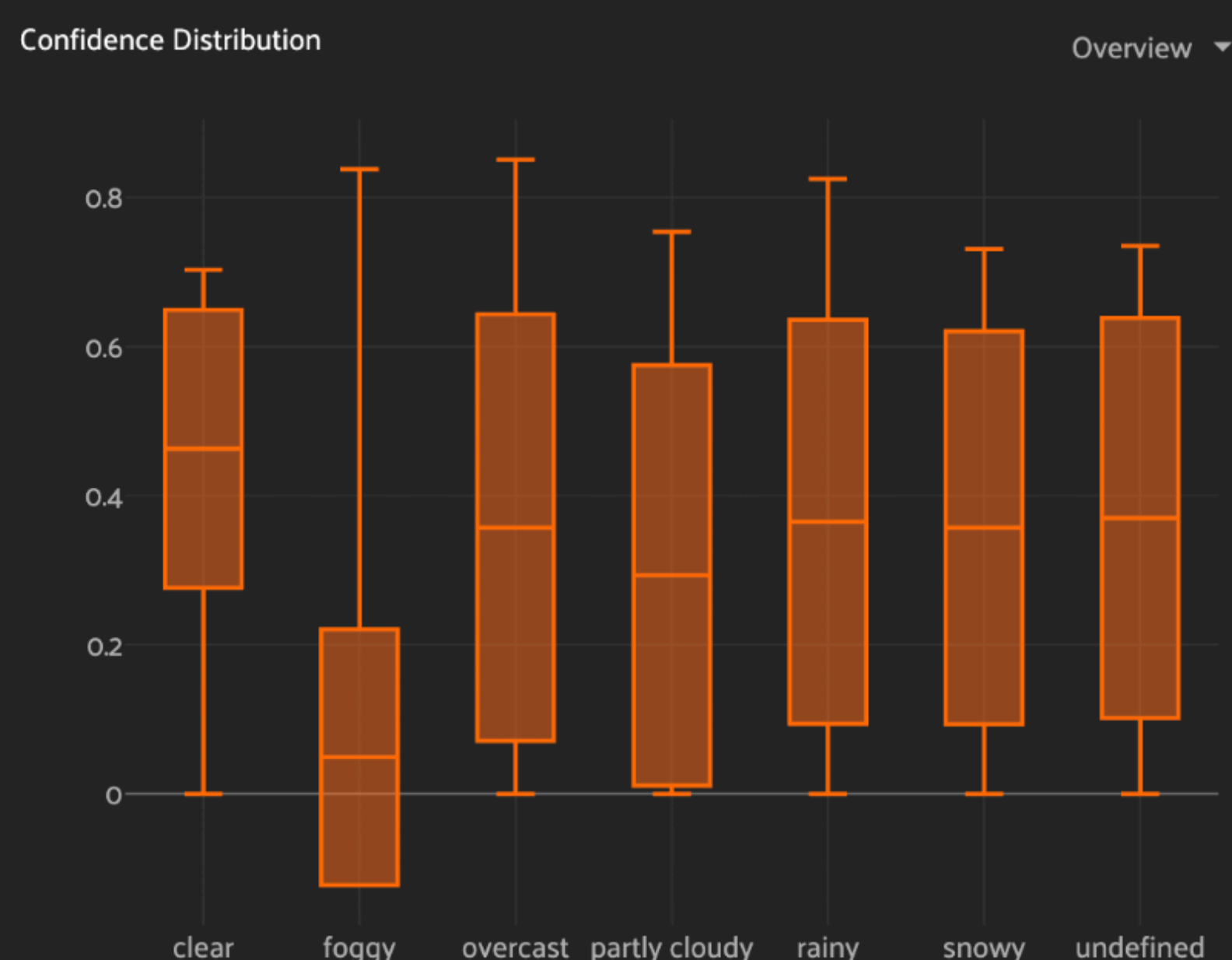
Track data drift and monitor model confidence to detect and address ongoing model failures. Production is where models face their ultimate test, with real-world data that constantly evolves.

Essential monitoring components

A good tool allows users to visualize and track data drift over time, compare model confidence with real-world outcomes, and detect overconfidence in incorrect predictions.



Data drift visualization showing how feature distributions shift over time in production.



Model confidence distribution.



Conclusion

Building a reliable computer vision model is a full-lifecycle endeavor. By examining common failure modes at each stage, we see a recurring theme: most issues are preventable with the right practices. Many model failure modes are "data-centric" rather than algorithm-centric.

- **Key Insight:**
Curating better datasets, evaluating models thoroughly, and implementing rigorous testing can significantly reduce the likelihood of your computer vision model failing in the field.

As the field advances, there's increasing recognition that spending effort on data curation, annotation, error mitigation, and monitoring yields huge dividends in reliability.

About Voxel51

Voxel51, the company behind FiftyOne, offers an end-to-end computer vision platform that helps organizations unlock the full potential of their data. By providing cutting-edge tools for visualizing, curating, and evaluating computer vision data and models, organizations building vision AI systems gain deep visibility into their datasets and model performance, critical for ensuring reliability in production.

By allowing teams to explore data and predictions side by side, FiftyOne helps pinpoint the root causes of failure and drive smarter, faster model improvements to reduce risks and accelerate the deployment of high-performing vision models.

For organizations, the takeaway is clear:

Successful CV projects require a blend of good engineering and good data science: get the data right, train the model right, test it rigorously, and monitor it in production.

The cost of failure is too high to leave to chance. The techniques in this guide provide essential safeguards for any organization serious about deploying reliable computer vision systems.

The platform makes it easy to identify labeling errors, edge cases, and distribution gaps through powerful visual exploration and analysis. It integrates with annotation workflows, supports both manual and automated labeling, and enables robust curation and model evaluation through metrics, embeddings, and scenario-based analysis.

Interested in learning more?

[Book a demo](#) with our team of experts to learn more on how you can build and deploy reliable visual AI models.



Maximize AI performance with better data

Voxel51 accelerates AI development by making it easier to explore, visualize, and understand both data and model performance.

Whether you're debugging a model or curating the right data to train it, Voxel51 gives you the clarity you need to move faster. Understand the true distribution and diversity of your data, surface failure modes and edge cases, and get to production sooner with auto-labeling that reduces cost by up to 100,000x.



Book a demo