ERROR TERRAIN ANALYSIS FOR MACHINE LEARNING: TOOL AND VISUALIZATIONS

Rick Barraza, Russell Eames, Yan Esteve Balducci, Josh Hinds, Scott Hoogerwerf, Eric Horvitz
Ece Kamar, Jacquelyn Krones, Josh Lovejoy, Parham Mohadjer, Ben Noah, Besmira Nushi
Microsoft (Redmond, WA, USA)

Introduction. Developing and maintaining reliable machine learning systems requires a deep understanding of system failures and rigorous evaluation processes (Sculley et al., 2015; Breck et al., 2016). Unfortunately, failure understanding and systematic evaluation for machine learning are not yet supported by traditional development tools (Holstein et al., 2018; Amershi et al., 2019). Aggregated and high-level evaluation methods such as single-score performance numbers or even multi-class confusion matrices often hide important conditions of failure (Nushi et al., 2018; Mitchell et al., 2018; Chouldechova & G'Sell, 2017). As recently noted by Buolamwini & Gebru (2018), a poor understanding of details of model underperformance by system engineers and end users can lead to high-stake mistakes.

In this demo, we present ongoing work to build an error analysis tool, which *helps engineers accelerate the development process by moving beyond aggregate scores to reveal a broader error terrain*. These efforts are based on a suite of recent research on systematic and rigorous error analyses aimed at more accountable and scrutable machine learning practices (Nushi et al., 2017; Lakkaraju et al., 2017; Nushi et al., 2018).

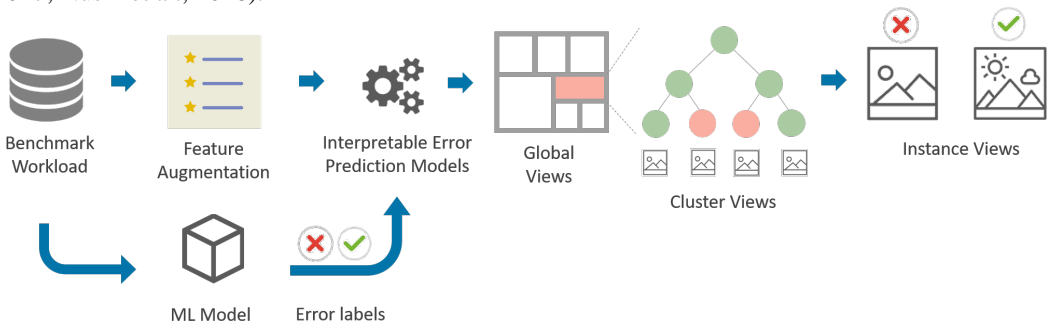

Figure 1: Error terrain analysis tool overview.

Tool description. Figure 1 shows an overview of the error analysis tool and the Appendix shows examples of its functions and visualizations. The first input of the tool is a benchmark dataset, which might be an external open-source benchmark, a dataset showing the performance of the deployed model in the real world, or an experimental diagnostic dataset built for debugging. The second input consists of individual model performance labels for each instance in the benchmark. We call these *error labels* and in their simplest form they denote whether the model has succeeded or failed for an instance. The last input includes a set of features that augment the benchmark data and will be used later for characterizing and predicting errors. These features might be harvested from external classifiers and APIs or internal model representations if the user has access to the model. In this study, we have collected sets of features that are augmented by inferences obtained via libraries and classifiers from the Azure Cognitive Services.

Based on the benchmark data and the augmented features, the tool trains interpretable models for predicting and describing the error labels and hence model performance. The tool extracts and displays hidden conditions of failure, explaining when and how the machine learning model of interest fails or succeeds. In this context, the tool offers three different views on model performance: *global views*, *cluster views*, and *instance views*. Global views break down performance based on known important sensitive pivots (e.g., demographics, locale identifiers etc.) or unsupervised clusters of the input domain. Cluster views use decision trees as interpretable models to describe errors for a given cluster or pivot grouping. Global views and cluster views can be unified if the user prefers to use a single global model for error analysis. Nevertheless, custom cluster views are convenient if the engineer is interested in diving deeper into problems that appear within a particular cluster or grouping. Finally, instance views enable visual analyses either for comparing two instances or for debugging a single instance.

The demo showcases these functionalities by using a real-world case study in the face recognition domain for gender detection. The main goal is to walk the audience through the process of error terrain analysis, explaining how such a process can facilitate quality control in machine learning and guide future data collection and model improvements. The following Appendix shows examples from the gender detection task.

REFERENCES

Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. Software engineering for machine learning. In *International Conference on Software Engineering, Software Engineering in Practice*, 2019.

Eric Breck, Shanqing Cai, Eric Nielsen, Michael Salib, and D Sculley. Whats your ml test score? a rubric for ml production systems. 2016.

Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency*, pp. 77–91, 2018.

Alexandra Chouldechova and Max G'Sell. Fairer and more accurate, but for whom? *arXiv preprint arXiv:1707.00046*, 2017.

Kenneth Holstein, Jennifer Wortman Vaughan, Hal Daumé III, Miro Dudík, and Hanna Wallach. Improving fairness in machine learning systems: What do industry practitioners need? *arXiv preprint arXiv:1812.05239*, 2018.

Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Eric Horvitz. Identifying unknown unknowns in the open world: Representations and policies for guided exploration. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. 2018.

Besmira Nushi, Ece Kamar, Eric Horvitz, and Donald Kossmann. On human intellect and machine failures: Troubleshooting integrative machine learning systems. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Besmira Nushi, Ece Kamar, and Eric Horvitz. Towards accountable ai: Hybrid human-machine analyses for characterizing system failure. In *Sixth AAAI Conference on Human Computation and Crowdsourcing*, 2018.

David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*, pp. 2503–2511, 2015.

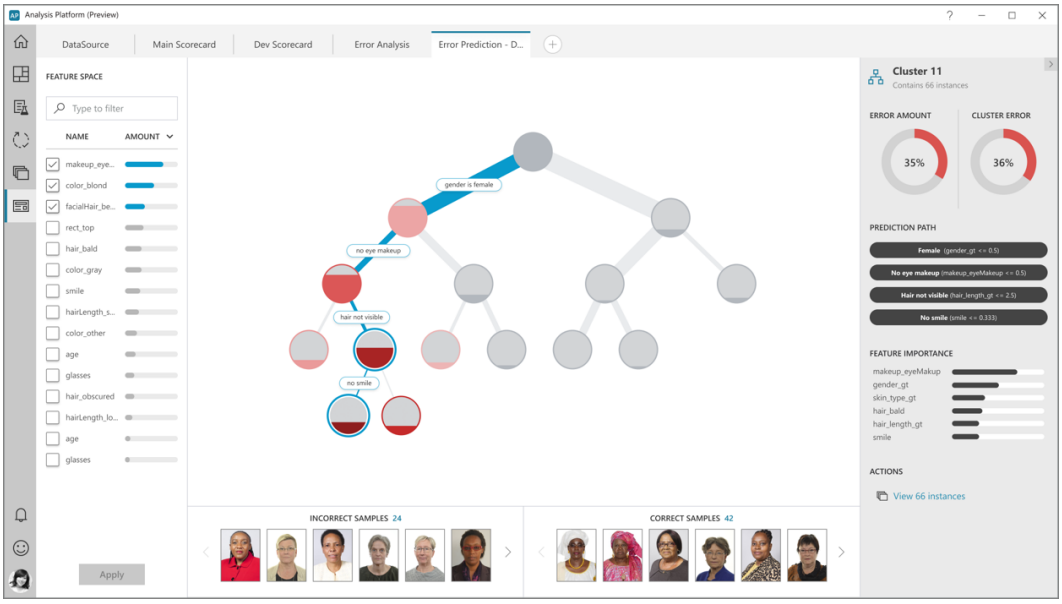

Figure 2: Error terrain analysis tool applied to analyze errors of a gender detector classifier. The global view here is a unified cluster view generated using the whole benchmark dataset. The panel on the left shows the set of augmented features generated via the FaceAPI classifiers in the Azure Cognitive Services, along with their importance for error prediction. The user can select which features should be used to generate the explanation tree. The right panel summarizes the decision tree conditions that constitute a selected node as well as the error for that particular node. The lower panel shows correct and incorrect instances from the selected node in the tree. In order to switch to an instance view, the user can pick particular samples from both sides and analyze and compare.

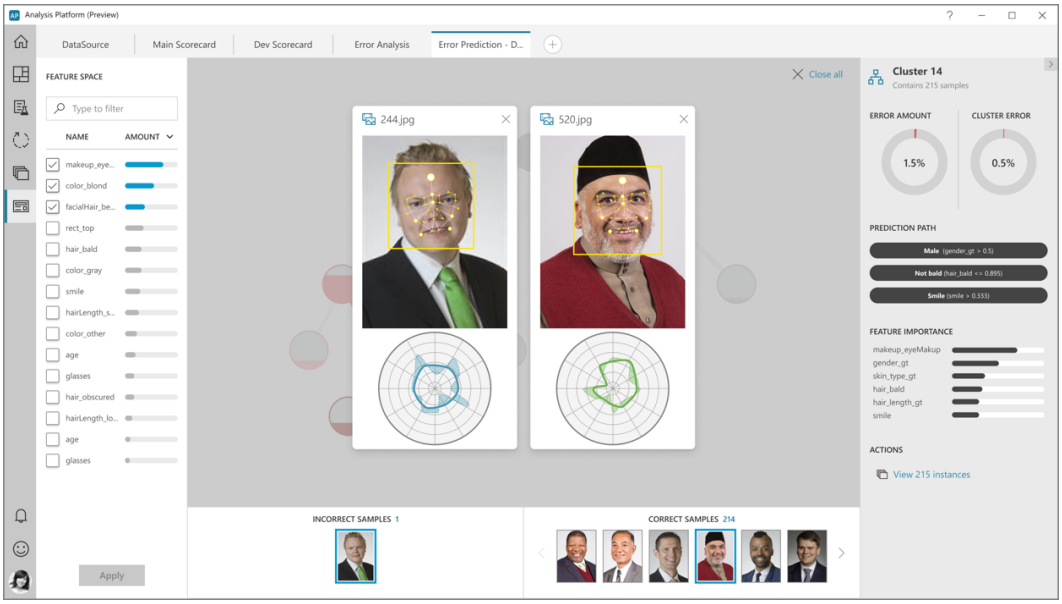

Figure 3: Example of an instance view showing face landmarks (e.g., eye, mouth, noise points) on two different instances, for which the gender detector is succeeding (right) or failing (left).

1 APPENDIX: CASE STUDY ON GENDER DETECTION

This appendix illustrates how the error terrain analysis tool can be applied to better understand errors of a machine learning model designed to detect gender in face images. This task received interest in industry and research, due to recent findings by (Buolamwini & Gebru, 2018) discovering that gender detector classifiers provided by various APIs perform significantly worse for women with a darker skin tone. This non-uniform error behavior is a typical example of how single-score evaluation strategies and quality control can hide important conditions of failure. Our results show that the proposed tool not only can discover such conditions but it can also enrich the analysis by introducing augmented features as error explainers.

Input. For this case study, we use a replication of the Pilot Parliaments Benchmark (PPB) using the same data collection process as in the GenderShades study (Buolamwini & Gebru, 2018). The dataset contains face images from parliament representatives across six countries around the world and is balanced with respect to gender and skin color. These pivots are available in the data and we use them as part of the feature augmentation. The other features were generated via FaceAPI classifiers available in the Azure Cognitive Services[1]. Among others, these classifiers detect hair color, emotion, makeup, accessories etc., and evaluate the image with respect to light exposure, noise, and blurring. We used the PPB dataset to evaluate an outdated gender detection model, which was also studied in the GenderShades study. The first objective was to check 1) whether these results could be reproduced in a time-efficient way for engineers, and 2) whether there existed more conditions of failure unknown to engineers.

Error Terrain Analysis. Figure 2 shows the tool user interface after the input data has been imported. The global view in the central panel here is a unified cluster view generated using the whole PPB dataset. The panel on the left shows the set of augmented features generated along with their importance for error prediction (information gain). The user can select which features should be used to generate the explanation tree. This flexibility with respect to feature selection is necessary if the user wants to explore different decision trees. For example, some of the features might be very well known to engineers and during exploration they might want to temporarily exclude them from the analysis so that other less important but still informative failure conditions can be highlighted.

Based on the selected features, the tool trains a decision tree with the error labels as target and summarizes the findings using conditional expressions on features. In other words, the tree uses the most conditionally informative features to separate correct instances from the incorrect ones. While doing so, the chain of <feature, operator, error_label> triples in the root-to-node path will serve as an explanation for a particular selected node. For example, the selected path in the tree in Figure 2 shows that the error rate increases from 5.5% (overall PPB) to 36% for women with no eye makeup, who have invisible or short hair and are not smiling. More details are shown in the right panel, including the error amount, which represents the percentage of errors that this node is responsible for with respect to the overall number of errors in the whole benchmark. These findings show that indeed the tool can simplify the process of error analysis and uncover unknown error patterns. The extracted critical paths (i.e., ending with red nodes) can be further used to guide engineering decisions on data collection, model improvement, and user interaction adjustments.

The lower panel shows correct and incorrect instances from the selected node in the tree. This division helps the user to further explore the data and possibly create further hypotheses for error explanation. For instance, in our case study, we initially did not have hair length as a feature. After exploring the data this way, we noticed that many women detected as men in the parent node had short hair (or invisible due to hair accessories). We hypothesized that the feature might be important for error analysis and therefore collected data for this particular feature via crowdsourcing, which resulted in the critical path including hair length as shown in Figure 2. Exploring the tree on the left branch of the node with hair length as a feature shows that the cluster error rate for women with no eye makeup is 30.8% if women have short or invisible hair, and 12.8% if women have long or medium length hair.

In order to switch to an instance view, the user can pick particular samples from both sides of the lower panel and analyze and compare. Figure 3 depicts an example of an instance view showing face landmarks (e.g., eye, mouth, noise points) on two different instances, for which the gender detector is succeeding (right) or failing (left). We are planning to focus our future research efforts on including visual explanations in the instance view that can highlight the most important counterfactual features, which differentiate successes from failures. At the same time, these features will also be leveraged as part of the initial feature augmentation.

[1] https://azure.microsoft.com/en-us/services/cognitive-services/