Towards Automated ML Model Monitoring: Measure, Improve and Quantify Data Quality

Tammo Rukat, Dustin Lange
[tammruka,langed]@amazon.com
Amazon Research

Sebastian Schelter
sebastian.schelter@nyu.edu
New York University

Felix Biessmann
fbiessmann@beuth-hochschule.de
Beuth University Berlin

1 INTRODUCTION

Machine learning (ML) has become a central component in modern software applications, giving rise to many new challenges [8, 15, 20]. Tremendous progress has been made in this context with respect to model serving [1, 6, 10], experiment tracking [14, 16, 22, 23], model diagnosis [4, 5, 11, 21] and data validation [4, 18].

In this paper, we focus on the arising challenge of automating the operation of deployed ML applications, especially with respect to monitoring the quality of their input data. Existing approaches [1, 18, 22] for this problem have not yet reached broad adoption. One reason for that is that they often require a large amount of domain knowledge, e.g., to define "data unit tests" and corresponding similarity metrics and thresholds for detecting data shifts. Additionally, it is very challenging to test data at early stages of a pipeline (e.g., during integration) without explicit knowledge of how the data will be processed by downstream applications. In other cases, the engineers in charge of operating a deployed ML model may not have access to the model internals, for example if they leverage a popular cloud ML service such as Google AutoML[1] for training and inference. Integrating and automating data quality monitoring into ML applications is also difficult due to the lack of agreed upon abstractions for defining and deploying such applications.

We list three approaches to tackle data quality in ML applications from recent work: (*i*) Measuring data quality with "data unit tests" using the *Deequ* [18] library; (*ii*) Improving data quality with missing value imputation using the *DataWig* [3] library; and (*iii*) Quantifying the impact of data quality issues on the predictive performance of a deployed ML model [19]. Finally, we outline challenges and potential directions for combining these approaches and for automating their configuration in real world deployment settings.

2 INDIVIDUAL APPROACHES

Successful ML applications require careful data selection, data preprocessing and model building. Given that the responsibilities for such applications are usually distributed across people with different competencies [12], engineers in charge of operating ML applications should be supported

[1]https://cloud.google.com/automl/

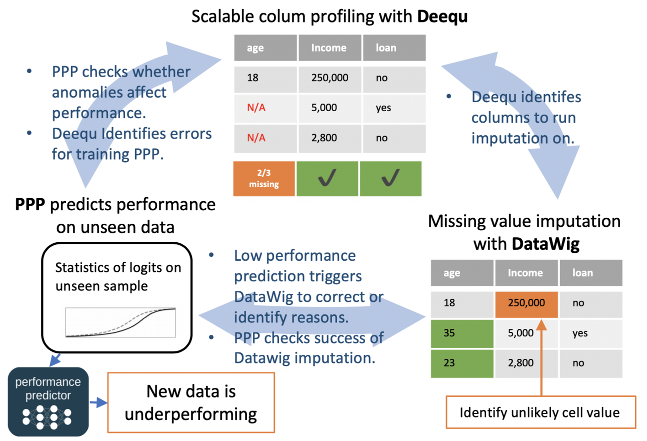
Scalable colum profiling with Deequ
PPP checks whether anomalies affect performance.
Deequ Identifies errors for training PPP.
Deequ identifes columns to run imputation on.
age
Income
loan
18
250,000
no
N/A
5,000
yes
N/A
2,800
no
2/3 missing
PPP predicts performance on unseen data
Statistics of logits on unseen sample
Low performance prediction triggers DataWig to correct or identify reasons.
PPP checks success of Datawig imputation.
Missing value imputation with DataWig
35
23
performance predictor
New data is underperforming
Identify unlikely cell value

Figure 1: Vison for integrating large-scale data quality verification with Deequ, statistical missing-value imputation, and pipeline performance prediction for *automatic* verification, "patching" and alarming of ML pipelines.

with automatic data quality monitoring infrastructure, which (*a*) does not require domain expertise, and (*b*) does not require access to model internals.

Measuring data quality. When confronted with poor predictive performance of a model, it is usually a good idea to first inspect its input data. We have previously proposed *Deequ*[2] [17, 18] for the definition and scalable execution of data quality tests. The library allows users to define quality tests for structured data in a declarative manner, and execute them on large datasets with Apache Spark. Parts of the model monitoring functionality of the cloud ML platform *Amazon Sagemaker* have been built upon Deequ[3].

Despite the growing adoption of Deequ, a number of challenges remain for further automation. In particular, it is difficult to automatically decide which statistics to inspect for which columns, and how to set thresholds for alarming based on these statistics. We are currently experimenting with an automation approach for use cases where a new batch of

[2]https://github.com/awslabs/deequ
[3]https://aws.amazon.com/blogs/aws/amazon-sagemaker-model-monitor-fully-managed-automatic-monitoring-for-your-machine-learning-models/

data needs to be tested and ingested at regular intervals (e.g., daily ingestion of log files). In this approach, we maintain time series of column statistics from previously observed data batches. To decide whether a new batch of data should be accepted, we compare its statistics to a forecast-based estimate of the expected statistics based on the observed time series.

Improving data quality. Once data quality issues are detected, they need to be addressed appropriately. We may for instance want to impute missing values. Libraries like *DataWig*[4] [2, 3] support missing value imputation for structured data in a fully automated manner. DataWig automatically featurizes all input columns, except the column with to-be-imputed values and trains a classifier on a concatenation of these features to predict the missing value. It automatically tunes hyperparameters in a given time budget, performs domain adaption by detecting and correcting shifts between rows that are used for training and rows that are subject to imputation, and calibrates likelihoods for imputed values allowing to control which imputed values are useful for downstream applications. These properties render it attractive for data engineers who do not have the necessary domain expertise and/or time to investigate every dataset in detail [7]. Exploiting dependencies between features is not only helpful for imputation, but also for monitoring quality on a single record level. Consider the example highlighted in [12], where the values for the column `country` change from upper case (`US`) to lower case (`us`), causing problems in downstream feature extractors. At serving time, a well-calibrated imputation model that treats these values as `missing` can identify the lower case version, (`us`), as unlikely under the previously learnt model and raise warnings automatically. A practical limitation of this approach, in particular with regard to automation, is the difficulty to deal with rare values that are common in heavy-tailed real-world datasets and with previously unseen values in general.

Quantifying the impact of data quality issues on the predictive performance of an ML model. A central question for data quality monitoring is the impact that any observed change in the distribution of the input data has on the predictive performance of the deployed ML model [12]. The approaches described above are model independent and, thus, have no means of quantifying this downstream effect. Imagine a scenario in which a model simply ignores a given input feature (e.g., due to L1 regularisation). A shift in this feature may be detected by systems monitoring the input data and might lead to unnecessary alarms. Recent approaches to model monitoring leverage statistics of the model predictions [9, 13] in a model agnostic way.

[4]https://github.com/awslabs/datawig

We have built upon this work to predict the performance of generic black box classifiers[5] [19]. We achieve this by generating perturbed copies of the original data where each copy resembles typical errors and data quality problems. Ideally, typical types of perturbations are provided by a domain expert. The original classifier applied to the perturbed datasets yields pairs of the form (*quantized output distribution*, *model performance*). These are fed to a regression model that learns to predict the model performance, the "pipeline performance predictor" (PPP). An alarm is raised at serving time if the predicted performance falls below a predefined performance threshold.

The only step requiring domain knowledge within this procedure is the specifiction of potential data errors. Our experiments, however, indicate that it is often sufficient to train the regression model with pre-specified common error types [4]. One shortcoming of this approach lies in its unawareness of the particular data points or properties that cause a predicted performance drop. Future work should put emphasis on a better understanding of the effect that errors in the data have on a model's output distribution.

3 VISION FOR INTEGRATION

While the three above described techniques each have their independent use case, fully automated data quality monitoring may greatly benefit from a tight integration. Consider, for instance, the following scenario: Deequ detects missing values or likely errors in a particular column of a dataset. Based on a DataWig model, values in this column are imputed or replaced. Eventually, PPP verifies that the errors would indeed have caused a large drop in model performance and that this drop is significantly mitigated with DataWig's corrections.

The effort of creating an appropriate configuration would merely be comprised of (i) defining column constraints in Deequ (which can be semi-automated), (ii) specifying typical errors (which can be dispensed with for common scenarios) and required model performance thresholds for PPP, (iii) setting a threshold confidence for missing-value imputation in DataWig. Then the entire process operates without human intervention: The ML pipeline is patched automatically and warnings/errors are raised when degrading performance cannot be repaired.

REFERENCES

[1] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, et al. 2017. Tfx: A tensorflow-based production-scale machine learning platform. *KDD*, 1387–1395.

[5]https://github.com/schelterlabs/learning-to-validate-predictions

[2] Felix Biessmann, Tammo Rukat, Phillipp Schmidt, Prathik Naidu, Sebastian Schelter, Andrey Taptunov, Dustin Lange, and David Salinas. 2019. DataWig: Missing Value Imputation for Tables. *Journal of Machine Learning Research* 20, 175 (2019), 1–6. http://jmlr.org/papers/v20/18-753.html

[3] Felix Biessmann, David Salinas, Sebastian Schelter, Philipp Schmidt, and Dustin Lange. 2018. Deep Learning for Missing Value Imputation in Tables with Non-Numerical Data. *CIKM*, 2017–2025.

[4] Eric Breck, Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2019. Data Validation for Machine Learning. In *Conference on Systems and Machine Learning (SysML). https://www. sysml. cc/doc/2019/167. pdf.*

[5] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. 2019. Slice finder: Automated data slicing for model validation. *ICDE*, 1550–1553.

[6] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J Franklin, Joseph E Gonzalez, and Ion Stoica. 2017. Clipper: A low-latency online prediction serving system. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*. 613–627.

[7] Julie Josse, Nicolas Prost, Erwan Scornet, and Gaël Varoquaux. 2019. On the consistency of supervised learning with missing values. arXiv:stat.ML/1902.06931

[8] Arun Kumar, Robert McCann, Jeffrey Naughton, and Jignesh M Patel. 2016. Model selection management systems: The next frontier of advanced analytics. *ACM SIGMOD Record* 44, 4 (2016), 17–22.

[9] Zachary C Lipton, Yu-Xiang Wang, and Alex Smola. 2018. Detecting and Correcting for Label Shift with Black Box Predictors. *ICML* (2018).

[10] Christopher Olston et al. 2017. Tensorflow-serving: Flexible, high-performance ml serving. *ML Systems workshop at NeurIPS* (2017).

[11] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated whitebox testing of deep learning systems. *SOSP*, 1–18.

[12] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2018. Data lifecycle challenges in production machine learning: a survey. *ACM SIGMOD Record* 47, 2 (2018), 17–28.

[13] Stephan Rabanser, Stephan Günnemann, and Zachary C Lipton. 2019. Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift. *NeurIPS* (2019).

[14] Cedric Renggli, Bojan Karlaš, Bolin Ding, Feng Liu, Kevin Schawinski, Wentao Wu, and Ce Zhang. 2019. Continuous integration of machine learning models with ease. ml/ci: Towards a rigorous yet practical treatment. *arXiv preprint arXiv:1903.00278* (2019).

[15] Sebastian Schelter, Felix Biessmann, Tim Januschowski, David Salinas, Stephan Seufert, and Gyuri Szarvas. 2018. On Challenges in Machine Learning Model Management. *Data Engineering* (2018).

[16] Sebastian Schelter, Joos-Hendrik Boese, Johannes Kirschnick, Thoralf Klein, and Stephan Seufert. 2017. Automatically tracking metadata and provenance of machine learning experiments. *Machine Learning Systems workshop at NeurIPS* (2017).

[17] Sebastian Schelter, Stefan Grafberger, Philipp Schmidt, Tammo Rukat, Mario Kiessling, Andrey Taptunov, Felix Biessmann, and Dustin Lange. 2019. Differential Data Quality Verification on Partitioned Data. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1940–1945.

[18] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. 2018. Automating Large-Scale Data Quality Verification. *Proceedings of the VLDB Endowment* 11, 12 (2018).

[19] Sebastian Schelter, Rukat Tammo, and Felix Biessmann. 2020. Learning to Validate the Predictions of Black Box Classifiers on Unseen Data. *SIGMOD* (2020).

[20] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, 2503–2511.

[21] Manasi Vartak, Joana M F da Trindade, Samuel Madden, and Matei Zaharia. 2018. Mistique: A system to store and query model intermediates for model diagnosis. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 1285–1300.

[22] Manasi Vartak, Harihar Subramanyam, Wei-En Lee, Srinidhi Viswanathan, Saadiyah Husnoo, Samuel Madden, and Matei Zaharia. 2016. Model DB: a system for machine learning model management. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. ACM, 14.

[23] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, et al. 2018. Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.* 41, 4 (2018), 39–45.