Learning to Validate the Predictions of Black Box Machine Learning Models on Unseen Data

Sergey Redyuk[*] Sebastian Schelter[†] Tammo Rukat[**]
Volker Markl[*] Felix Biessmann[‡]
{sergey.redyuk,volker.markl}@tu-berlin.de sebastian.schelter@nyu.edu {tammorukat,felix.biessmann}@gmail.com
[*]Technische Universität Berlin [†]New York University [**]University of Oxford [‡]Beuth University Berlin

ABSTRACT

When end users apply a machine learning (ML) model on new unlabeled data, it is difficult for them to decide whether they can trust its predictions. Errors or shifts in the target data can lead to hard-to-detect drops in the predictive quality of the model. We therefore propose an approach to assist non-ML experts working with pretrained ML models. Our approach estimates the change in prediction performance of a model on unseen target data. It does not require explicit distributional assumptions on the dataset shift between the training and target data. Instead, a domain expert can declaratively specify typical cases of dataset shift that she expects to observe in real-world data. Based on this information, we learn a *performance predictor* for pretrained black box models, which can be combined with the model, and automatically warns end users in case of unexpected performance drops. We demonstrate the effectiveness of our approach on two models – logistic regression and a neural network, applied to several real-world datasets.

ACM Reference Format:
Sergey Redyuk, Sebastian Schelter, Felix Biessmann, Volker Markl. 2019. Learning to Validate the Predictions of Black Box Machine Learning Models on Unseen Data. In *Workshop on Human-In-the-Loop Data Analytics (HILDA'19), July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3328519.3329126

HILDA'19, July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-6791-2/19/07…$15.00
https://doi.org/10.1145/3328519.3329126

1 INTRODUCTION

New challenges emerge with the advent of machine learning (ML) as a central component in modern information infrastructures [4]. Problems in this space originate, for example, from developers having to handle unexpected shifts in the data that is fed into systems powered by machine learning. Consider the following examplary scenario: A consultant with ML expertise creates a ML model for predicting the sales numbers of competitor products, and provides the final trained model to an engineering team from an e-commerce company. Imagine that some time later, this engineering team accidentally introduces errors in the web crawling code that fetches new target data for the model, and the error changes the scale of certain numeric attributes. The model will still output numeric predictions for unseen competitor products, but the guarantees about the reliability of these predictions are lost due to the unexpected shift in the data. Unfortunately, it is really difficult for the engineers to validate the predictions, as there is no automated way to retrieve the ground truth (the future sales numbers of the competitor in this scenario).

While ML experts have specialized knowledge to debug models and predictions in such cases, *there is a lack of automated, declarative tools for non-ML expert users to help them decide whether they can trust the predictions of a ML model on new data.*

We therefore propose an approach where domain experts and end users only need to declaratively specify types of potential shifts in the target data of their ML models. Based on these, we describe how to learn a *performance predictor* for an existing ML model, which can be shipped together with the model (Section 2). This predictor empowers end users, as it can automatically warn them in case of potential drops of the predictive quality of the model on unseen data. Our method is easy to implement, and provides several advantages over existing work (Section 3) in this area: (*i*) it does not require explicit distributional assumptions on the dataset shift; (*ii*) our approach works with general black box models, which consume raw input data (e.g., relational tuples or unstructed data, such as text or images), and hide model

details and internal feature representations. Examples for such black box models are scikit-learn pipelines, SparkML pipelines, or hosted models in cloud ML services.
The contributions of this paper are the following:

- We present an approach to automatically assess changes in the prediction scores of black box ML models on unseen, unlabeled and potentially erroneous target data (Section 2).
- We validate our approach on various datasets, shifts and models, and find that it predicts the true model score with an MAE of 0.01 in the majority of cases (Section 4).

2 APPROACH

Problem statement. We focus on generic black box ML models in the form of a classification function $f : \mathcal{X} \rightarrow \mathbb{R}^c$, which takes an example $\mathbf{x} \in \mathcal{X}$ and returns a prediction $\hat{\mathbf{y}} \in \mathbb{R}^c$ for the c classes. Here, $\mathbf{x}$ refers to raw input data for the black box model, e.g., a row from a relational table or dataframe, which can contain numerical, ordinal or categorical variables, as well as unstructured data, such as text and images. The black box model f will internally apply transformations to convert the raw input data to numerical features. The predictions $\hat{\mathbf{y}} \in \mathbb{R}^c$ are numeric vectors, and each dimension of $\hat{\mathbf{y}}$ corresponds to a probability of the c classes. We refer to the data that is available for the ML model f at training time as the *source dataset* – $\mathcal{D}_{\text{source}}$ of labeled examples $(\mathbf{X}, \mathbf{y})$ where $\mathbf{y}$ denotes the ground truth labels for the examples $\mathbf{X}$ from c classes. During training, an ML expert tunes all parameters and hyperparameters of the black box model f to maximize a score $\ell = L(f(\mathbf{X}), \mathbf{y})$ such as accuracy, precision, or area under the ROC curve *(AUC)*.

We consider a typical use case, where an end user (who is not an ML expert) applies the trained black box model to another dataset $\mathcal{D}_{\text{target}}$, referred to as *target dataset*. This data is not available at training time and has no labels available at prediction time. Most ML models rely upon the *i.i.d. assumption*, meaning that both the source and target data are sampled independently from the same stationary distribution. In real-world scenarios however, $\mathcal{D}_{\text{target}}$ could be corrupted (and thereby violate this assumption) due to dataset shift, e.g., caused by errors induced by the data preparation code, due to a distribution shift caused by changes in the underlying data generating process in the real-world, or due to adversarial attacks. In such cases, there is no guarantee about the reliability of the predictions. If the trained model is applied to target data that underwent a dataset shift, the resulting predictions can be arbitrarily wrong in principle. However, the predictive performance of the model, measured by the score $\ell_{\text{target}} = L(f(\mathbf{X}_{\text{target}}), \mathbf{y}_{\text{target}})$ could only be computed if we had access to the true labels $\mathbf{y}_{\text{target}}$, which are unknown for $\mathcal{D}_{\text{target}}$.

Algorithm 1 Training the performance predictor h.

Input: $(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}})$ disjunct partitions of $\mathcal{D}_{\text{source}}$
black box model f pretrained on $\mathcal{D}_{\text{train}}$

1: $(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) \leftarrow \mathcal{D}_{\text{test}}$
2: $\ell_{\text{test}} \leftarrow L(f(\mathbf{X}_{\text{test}}), \mathbf{y}_{\text{test}})$
3: $M \leftarrow \emptyset$
4: **for** shift generator $e \leftarrow E$ **do**
5: $\quad \mathbf{X}_{\text{corrupt}} \leftarrow$ corrupt $\mathbf{X}_{\text{test}} \in \mathcal{D}_{\text{test}}$ with e
6: $\quad \hat{\mathbf{Y}}_{\text{corrupt}} \leftarrow f(\mathbf{X}_{\text{corrupt}})$
7: $\quad \ell_{\text{corrupt}} \leftarrow L(\hat{\mathbf{Y}}_{\text{corrupt}}, \mathbf{y}_{\text{test}})$
8: $\quad \phi_{\text{corrupt}} \leftarrow \text{percentiles}(\hat{\mathbf{Y}}_{\text{corrupt}})$
9: $\quad M \leftarrow M \cup (\phi_{\text{corrupt}}, \ell_{\text{corrupt}})$
10: **end for**
11: $h \leftarrow$ train a regression model on M
12: **return** (h, l_{test})

The problem that we address in this work is to compute an estimate $\hat{\ell}_{\text{target}}$ of the score ℓ_{target}, which the black box model achieves on the target examples $\mathbf{X}_{\text{target}}$ without access to the corresponding ground truth labels $\mathbf{y}_{\text{target}}$. If the score drops, the model should not be used, and an ML expert should be consulted to decide what to do next.

Our approach is based on the idea of (i) simulating dataset shifts, specified by a domain expert, which could occur in the target data of a trained model, and (ii) subsequently learning a model-specific *performance predictor* (in the form of a regression model) to estimate the impact of these shifts on the black box model's performance. We simulate dataset shifts on a held-out partition of the source dataset for which we know the ground truth labels. If the user-specified shifts capture the dataset shifts encountered in practice well, then the outputs of our predictor will be a reliable estimate of the performance of the black box model on the new data at hand, even though we do not know the true labels.

Training the performance predictor. The steps for training a performance predictor h for a particular black box model are shown in Algorithm 1. We assume that we are provided with a black box model f which has been trained on the source data. Furthermore, a domain expert specifies a set of shift generators E that produce certain types of randomized dataset shifts. We apply each shift generator e to held-out data $\mathcal{D}_{\text{test}}$ from the training process, and thereby generate corrupted examples $\mathbf{X}_{\text{corrupt}}$ in line 5. We compute the actual prediction score ℓ_{corrupt} by comparing the model predictions $\hat{\mathbf{Y}}_{\text{corrupt}}$ with the true labels $\mathbf{y}_{\text{test}}$ in line 7. Existing work suggests that the univariate distributions of the output dimensions of the matrix $\hat{\mathbf{Y}} = f(\mathbf{X})$ are predictive of dataset shifts [2, 3]. We build on these findings, and leverage a univariate non-parametric estimate of the distributions of each output dimension of f. More concretely, we compute

several percentiles ϕ_{corrupt} of the black box model outputs $\hat{\mathbf{Y}}_{\text{corrupt}} = f(\mathbf{X}_{\text{corrupt}})$ as features for the performance predictor h. We record both the features ϕ_{corrupt} and the score as training data for our regression model in the set M (lines 3-10). Finally, we train our performance predictor h on these examples in line 11 to predict the score ℓ_{corrupt} from the features ϕ_{corrupt}.

Performance prediction on unseen target data. Algorithm 2 gives an example of how to apply our performance predictor h to unseen (and unlabeled) target data $\mathbf{X}_{\text{target}}$. First, we compute the required features ϕ_{target} in the form of percentiles of the outputs $\hat{\mathbf{Y}}_{\text{target}}$, which the model f assigns to the examples $\mathbf{X}_{\text{target}}$. Next, we have our performance predictor h estimate the score $\hat{\ell}_{\text{target}} = h(\phi_{\text{target}})$ of f on the target data, and compare this to the expected generalization error ℓ_{test}, which we computed during predictor training. If the difference exceeds a user-defined threshold t (e.g., 5%), we issue a warning to the end user.

Algorithm 2 Prediction validation for unlabeled target data.

Input: target data $\mathbf{X}_{\text{target}}$, black box model f, test score ℓ_{test}, performance predictor h, threshold t

1: $\hat{\mathbf{Y}}_{\text{target}} \leftarrow f(\mathbf{X}_{\text{target}})$
2: $\phi_{\text{target}} \leftarrow \text{percentiles}(\hat{\mathbf{Y}}_{\text{target}})$
3: $\hat{\ell}_{\text{target}} \leftarrow h(\phi_{\text{target}})$
4: **return** $(|\ell_{\text{test}} - \hat{\ell}_{\text{target}}| \,/\, \ell_{\text{test}}) \leq t$

3 RELATED WORK

Applying data management techniques in the ML space is a field with growing interest in recent years [4]. Several solutions were proposed for validating ML models and their predictions. Most of these originate from a statistical ML or a data management perspective. The former require assumptions on the nature of the dataset shift [2, 6], which often seem inapt to describe practically relevant data changes for engineers and domain experts, and the proposed methods often limit themselves to adapting a particular model or learning paradigm. Approaches from the data management community [1, 5] often require manual tuning, detailed knowledge of model internals and features, or do not directly adress the problem of validating model predictions. For instance, Google's TFX platform [1] offers skew detection capabilities for featurized input data, but forces the end user to manually select an appropriate distance function and threshold for particular features. In short, these methods are not applicable in scenarios where we intend to handle pretrained black box models that consume raw input data.

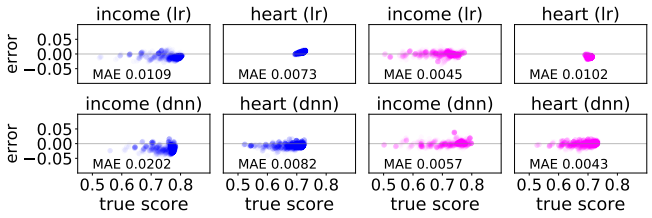
4 EVALUATION

Datasets and models. We experiment on three publicly available datasets from different domains. The *income*[1] dataset contains 48,842 records about adult income data, and the target variable denotes whether a person earns more than 50K dollars per year or not. The *heart*[2] dataset contains 70,001 records about cardiovascular diseases, and the target variable denotes the presence of a heart disease. The *tweets*[3] dataset comprises of 20,002 tweets, and the target variable denotes whether or not a tweet has trolling character (e.g., intended to insult someone). We leverage two popular classification approaches as black box models for our experiments: (*i*) we train a logistic regression model from scikit-learn (refered to as *lr*), using five-fold cross-validation where we grid search over regularization type and constant; (*ii*) we train a feed-forward neural network from tensorflow (refered to as *dnn*), comprised of two hidden layers with ReLU activation and a softmax output. We again use five-fold cross-validation and grid search over the size of the hidden layers.

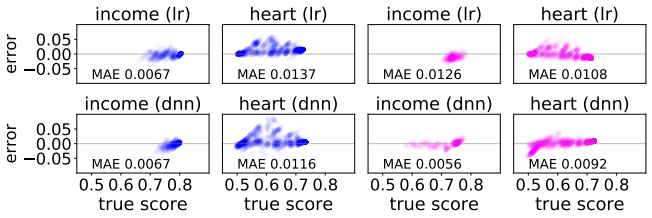
Setup. For every experimental run, we randomly partition a dataset into one partition with 90% of the records designated as *source data* and another disjoint partition with 10% of the records designated as *target data*. Next, we train one of our black box models on the source data, as described in Section 2. We implement our performance prediction approach in Python with a `RandomForestRegressor`, and train a predictor for each black box model, according to Algorithm 1. Afterwards, we apply shift generators to the unseen target data and create randomly corrupted versions of that data. We then have the performance predictor estimate the score on the unseen corrupted data, and compare this estimate to the actual score using the mean absolute error (*MAE*) (which we can compute in this virtual setup because we have access to the true labels). We run experiments to predict both accuracy as well as AUC; for experimental runs measuring the accuracy, we re-sample the data to have balanced classes to make the scores easier to interpret.

Types of errors and dataset shifts. We experiment with four different types of errors that we introduce into the unseen target data. For each type of error, we subsequently corrupt 0%, 5%, 25%, 50%, 75% and 99% of records in the dataset, and repeat each configuration 100 times: (*i*) *Missing values in categorical attributes – Figure 1(a):* In our first experiment, we choose a random set of categorical columns, and introduce missing values at random into these columns. (*ii*) *Gaussian shift in numeric attributes – Figure 1(b):* We randomly choose a set of numeric columns, and corrupt a

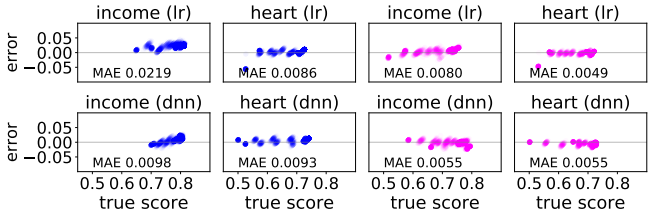
[1] https://archive.ics.uci.edu/ml/datasets/adult
[2] https://www.kaggle.com/sulianova/cardiovascular-disease-dataset
[3] https://dataturks.com/projects/abhishek.narayanan/Dataset%20for%20Detection%20of%20Cyber-Trolls/



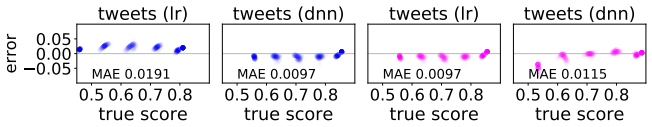
(a) Score estimation in presence of missing values in categorical attributes.



(b) Score estimation in presence of Gaussian shift in numeric attributes.



(c) Score estimation in presence of swapped attributes.



(d) Score estimation in presence of adversarial trolling tweets.

Figure 1: Score estimation for various shifts, models and datasets. The x-axis denotes the true model scores ℓ and the y-axis shows the corresponding absolute errors $\ell - \hat{\ell}$ from the estimate $\hat{\ell}$ given by our performance predictor. Each plot group shows accuracy prediction in blue on the left side and AUC prediction in magenta on the right.

fraction of their values by adding Gaussian noise centered in the data point with a standard deviation randomly scaled from the interval of 2 to 5. (*iii*) *Swapped column values – Figure 1(c):* We choose different pairs of categorical and numerical columns, and randomly swap the contained values between the columns. (*iv*) *Adversarial attack – Figure 1(d):* This experiment leverages the *tweets* dataset exclusively and simulates an adversarial attack. We assume that the authors of the trolling tweets try to fool our classifier by changing the spelling of their tweets, e.g., by converting the string "hello world" into "h3110 w041d".

Results. Figure 1 demonstrates the results of the corresponding experiments. We report the accuracy and AUC score that our approach provided for our two black box models – logistic regression and the neural network – on the datasets and the introduced shifts. Our approach predicts the true performance score within an MAE of 0.01 in the majority of cases. In three cases (logistic regression on the *income* data with swapped columns, application of the neural network on the *income* data with missing values, and logistic regression on the adversarial attack in the *tweets* dataset), our approach underpredicts the accuracy of the black box model, and only achieves an MAE of 0.02. In general, our predictions are close to the true performance score on the unseen data. We thereby enable end users to distinguish catastrophic model failures (such as cases with a low true score, where the model tends to randomly guess) from cases where the performance is only slightly affected by the shifts in the data.

5 FUTURE WORK

In future work, we plan to extend our approach to give more elaborate feedback to end users to empower them to fix the data errors and arrive at reliable model predictions. We will furthermore investigate how well our method performs when combinations of different shifts and errors are present in the target data, and how well our predictor works in cases where it has not been trained on the correct type of shift.

This work was supported by the Moore-Sloan Data Science Environment at New York University, as well as the BZML Berlin, ECDF and HEIBRiDS Graduate School Berlin.

REFERENCES

[1] Denis Baylor, Eric Breck, Heng-Tze Cheng, and Martin Zinkevich. 2017. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. *KDD* (2017), 1387–1395.

[2] Zachary C Lipton, Yu-Xiang Wang, and Alex Smola. 2018. Detecting and Correcting for Label Shift with Black Box Predictors. *arXiv preprint arXiv:1802.03916* (2018).

[3] Stephan Rabanser, Stephan Günnemann, and Zachary C Lipton. 2018. Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift. *arXiv preprint arXiv:1810.11953* (2018).

[4] Sebastian Schelter, Felix Biessmann, Tim Januschowski, David Salinas, Stephan Seufert, and Gyuri Szarvas. 2018. On Challenges in Machine Learning Model Management. *IEEE Data Engineering Bulletin* (12 2018).

[5] Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. 2018. Automating large-scale data quality verification. *PVLDB* 11, 12 (2018), 1781–1794.

[6] Masashi Sugiyama, Neil D Lawrence, Anton Schwaighofer, et al. 2017. *Dataset shift in machine learning*. The MIT Press.