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

Sergey Redyuk
TU Berlin
sergey.redyuk@tu-berlin.de

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

Tammo Rukat
University of Oxford
tammorukat@gmail.com

Volker Markl
TU Berlin
volker.markl@tu-berlin.de

Felix Biessmann
Beuth University Berlin
felix.biessmann@gmail.com

We propose an approach for non-ML experts to check whether they can trust the predictions of a black box ML model on unseen data

Problems and Challenges

- **Hard-to-trust predictions** of black box ML models
- **Drops in quality** of the model predictions, due to unexpected **errors / shifts** in the data

Reliability of Machine Learning Models

- Most of the ML models operate under the i.i.d assumption
- **No reliability guarantees** if the assumption is violated
- Reasons
  - **errors** in data preprocessing code
  - **changes** in the data generating process
- Hard to detect in practice

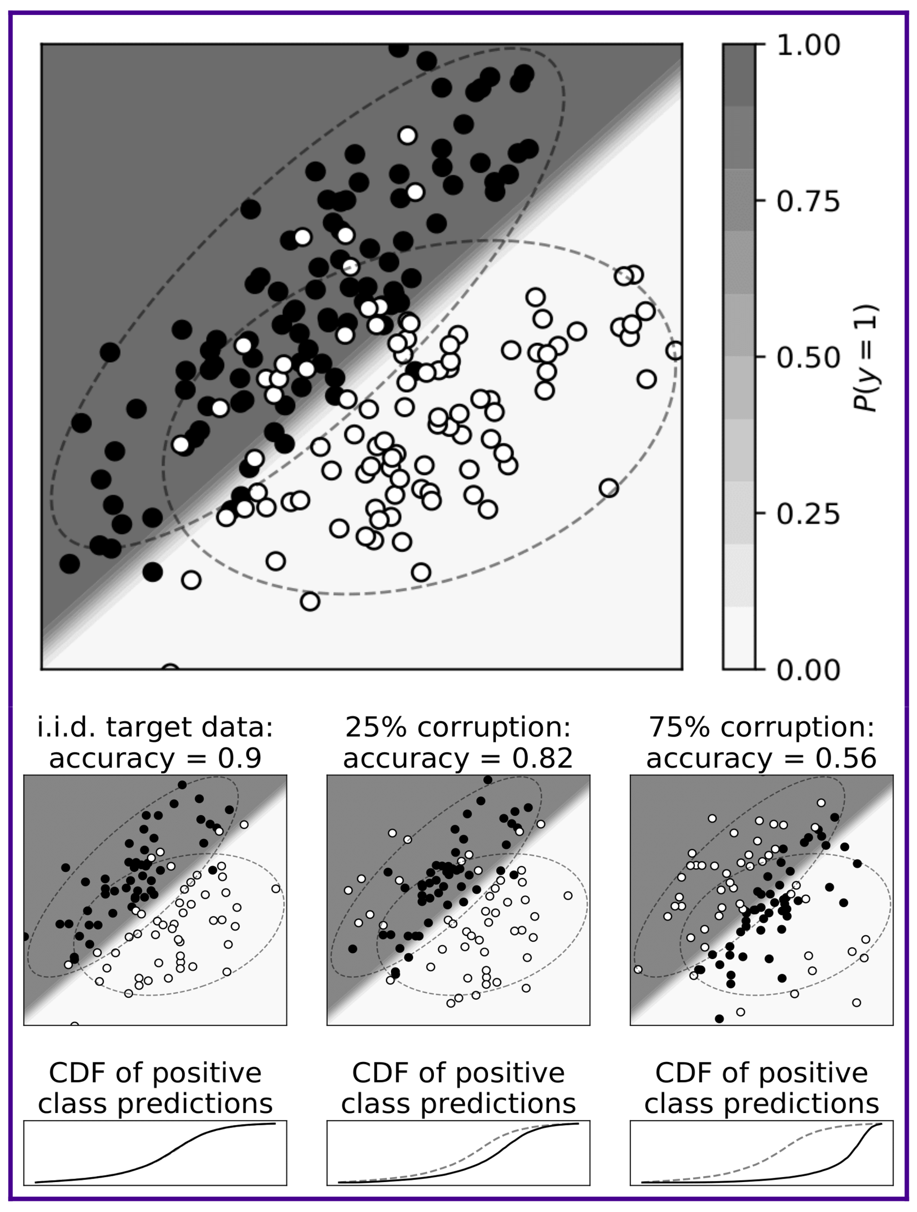
Manual Monitoring

ML experts

- make distributional **assumptions** about the source and target data
- apply **specialized** learning **algorithms**

Non-ML experts

- software / devops engineers, ETL designers, BI analysts, etc.
- need **automated** solutions that do not require expertise in statistics



Approach

We (a) **learn to predict the performance** of a pretrained black box ML model on unseen target data, given the type of the error (e.g., missing values, numeric outliers), and
(b) **raise alerts** if the predicted-vs-real performance **mismatch** is **detected**

1. **Domain expert / end user** declaratively **specifies** the expected types of **errors**
2. We generate **synthetically corrupted test data**
3. We apply the **black box ML model** on the corrupted data
4. We record (a) the **performance** of the model (e.g., accuracy), and
   (b) the **shape*** of the model's output
5. We **train a regression model** to predict the performance of the back box ML model

```
# Introduce missing values
class MissingValues(ErrorGen):
 def corrupt(data, prob)
  for row in data:
    if random() < prob:
      row[self.column] = NA

# Scale values
class Scaling(ErrorGen):
  def corrupt(data, prob):
    for row in data:
      if random() < prob:
        row[self.column] *= self.factor
```

* descriptive statistics over the model's outputs (e.g., **percentiles**)

Experimental Results

Two black box **ML models** – logistic regression, neural network, **four** introduced **shifts**, **three datasets**

MAE of 0.01 in the majority of cases

We distinguish cases with slight drops in performance from "catastrophic failures"

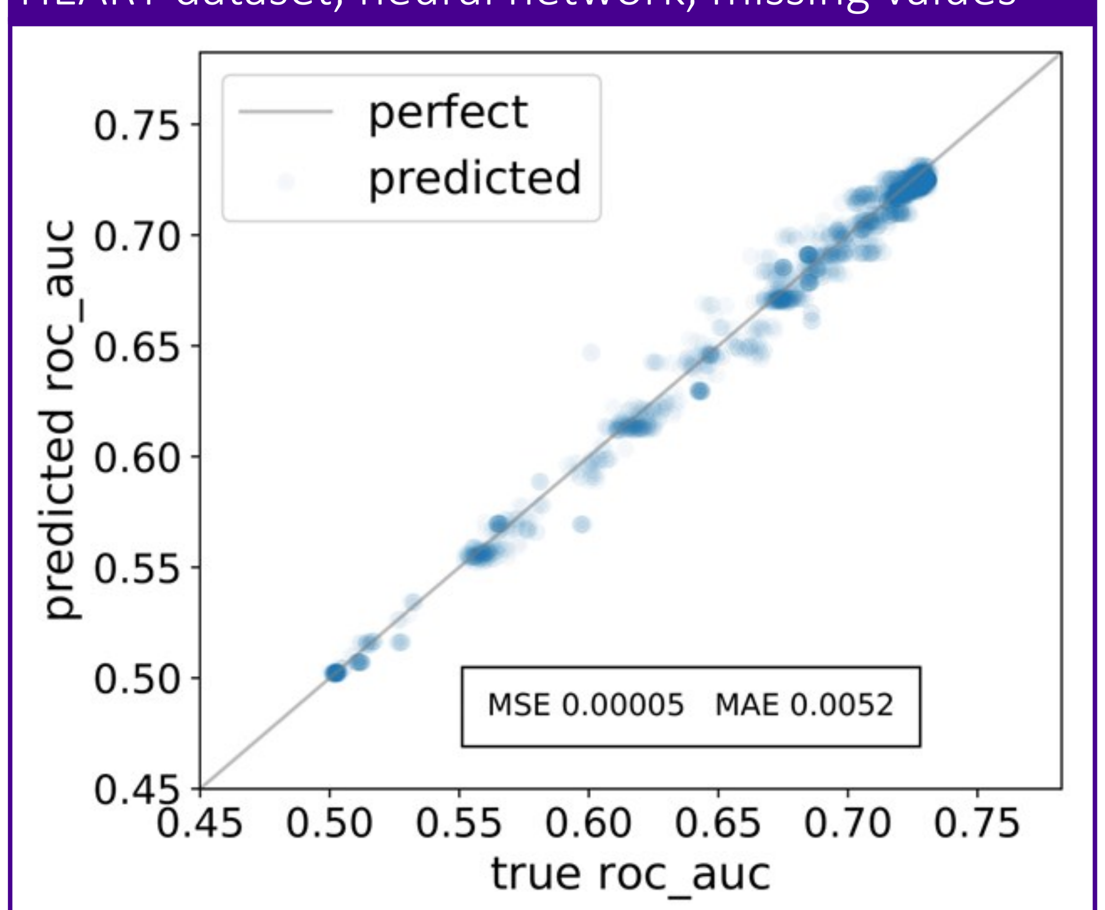
Advantages

- no distributional assumptions on the dataset shift
- applicable to general black box models that consume raw input data

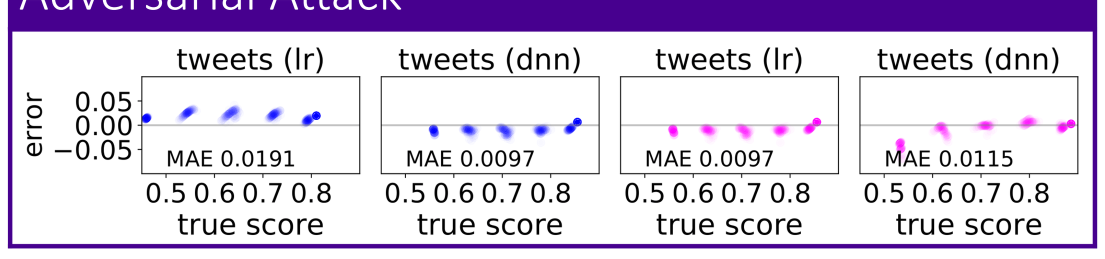
Future Work

- **Elaborate feedback** to end users
- Performance on **combinations of** different **shifts** and **errors**
- Performance on **not-yet-seen types of shift**

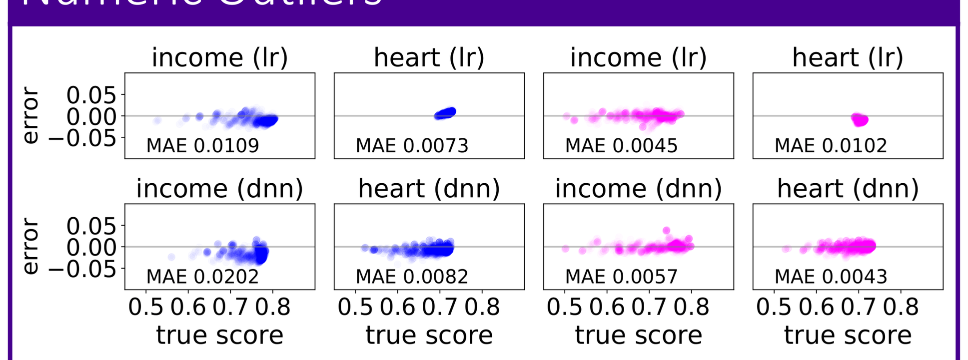
HEART dataset, neural network, missing values



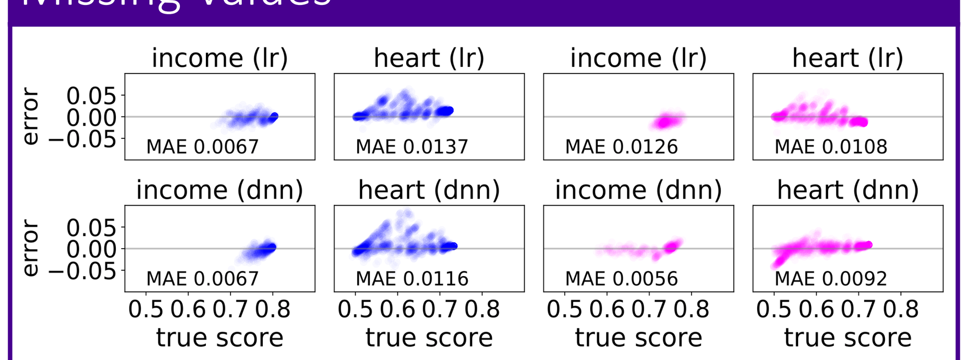
Adversarial Attack



Numeric Outliers



Missing Values



Swapped Values