UNITED NATIONS ECONOMIC COMMISSION FOR EUROPE

CONFERENCE OF EUROPEAN STATISTICIANS

Expert meeting on Statistical Data Confidentiality

26–28 September 2023, Wiesbaden

COACH: COmputer-Assisted output CHecking with Human-in-the-Loop

Manel Slokom*, Jel Vankan*, Peter-Paul de Wolf*, Martha Larson**

* Statistics Netherlands[1], ** Radboud University

m.slokom@cbs.nl

Abstract

In this paper, we introduce COACH (COmputer-Assisted output CHecking with Human-in-the-loop). COACH combines the power of machine learning models with human checkers' expertise to semi-automate output checking. By semi-automating output checking, we aim to facilitate the task of human checkers to check and decide whether output produced by a researcher using unprotected data from a statistical institute (e.g. in a Research Data Center) is safe for release to the public or not. First, we reproduce Domingo-Ferrer and Blanco-Justicia (2021) by leveraging simulated data and exploring diverse machine-learning algorithms. Then, our contributions extend prior work, by evaluating simulated data on real-world use cases and proposing pre-processing techniques to align simulated and real data. Next, our approach COACH empowers human checkers to review model predictions, providing valuable feedback to retrain and enhance model performance. Also, by utilizing global and local explainable AI method, COACH gains insights into model decision-making and influential variables. Our proposed approach revolutionizes output validation, fostering reliable and interpretable models for data-driven decision-making with human-in-the-loop interactions.

[1]The views expressed in this paper are those of the authors and do not necessarily reflect the policy of Statistics Netherlands.

1 Introduction

At the age of data-driven decision-making, researchers and policymakers more and more often make use of microdata made available by statistical institutes. Due to legal restrictions and privacy issues, their output needs to be checked for disclosure. Traditional output-checking methods at statistical agencies are heavily based on manual inspection, which is time-consuming and resource-intensive.

To address these challenges, we present COACH (COmputer-Assisted output CHecking with Human-in-the-loop), an approach that semi-automates output checking by incorporating human checkers expertise. Our work extends the research by Domingo-Ferrer and Blanco-Justicia (2021) exploring the use of various machine learning algorithms and pre-processing techniques on simulated data to achieve robust performance when applied to real-world use cases.

Automating Output Checking: The aim of automating output checking is to build machine learning models capable of predicting whether an output file is safe for release or not. Few existing works looked at automating output checking. Green et al. (2021) proposed a new toolkit for *automatic checking of research outputs* (for short called "ACRO"). Domingo-Ferrer and Blanco-Justicia (2021) proposed an approach that leverages machine-learning to assist human checkers in output checking. First, Domingo-Ferrer and Blanco-Justicia (2021) created simulated output checking data based on a subset of rules, called *14 rules-of-thumb* (we point readers to Bond et al. (2013) for more information about rules-of-thumb). Next, Domingo-Ferrer and Blanco-Justicia (2021) trained and tested how well the rules that are used to generate simulated data have been learned by a neural network. By leveraging simulated data and advanced machine learning algorithms, the models learn complex patterns and achieve high prediction accuracy. However, the application of such models to real-world use cases often poses challenges due to differences between simulated and real data distributions. To address this, COACH proposes a pre-processing step that allows to evaluate the machine learning model that is trained on simulated data to be test on real data.

The COACH Approach: Our approach emphasizes the importance of human-in-the-loop interactions to improve the accuracy and transparency of output checking. The COACH App serves as a collaborative platform for human checkers to review model predictions and provide valuable feedback. By incorporating human judgment, the COACH approach enhances the model validation process, ensuring outputs are reliable and safe for end-users.

We summarize the contributions of the paper as follows:

- Reproducing Domingo-Ferrer and Blanco-Justicia (2021)'s Work: We build on the foundation of Domingo-Ferrer and Blanco-Justicia (2021)'s research by exploring the performance of various machine learning algorithms when trained and tested on simulated data. This extension allows us to assess the efficacy of these models in real-world scenarios and highlights the need for adaptations to bridge the gap between simulated and real data distributions.
- Proposing pre-processing step: To address the discrepancies between simulated and real data distributions, we propose a pre-processing step. This step helps to evaluate a model trained on simulated data to be tested on real data.
- Evaluation of Simulated Data versus Real Test Data: Our study evaluates the performance of machine learning models trained on simulated data when applied to real test data. Through rigorous experimentation, we analyze the models' robustness, identifying areas of improvement for practical applications.
- Incorporating Human-in-the-Loop with COACH: We introduce COACH, an approach that integrates human-in-the-loop feedback in output checking. Through a user-friendly app, human checkers can confirm or reject the model's decisions (safe or not safe) and provide explanations when disagreements occur. The feedback obtained from human checkers is then utilized to retrain the machine learning models, improving their performance and adaptability.
- Utilizing Global and Local SHAP Values: To enhance the interpretability of our models, we employ global and local SHAP values (SHapley Additive exPlanations) to explain the model's output predictions. This provides insights into the model's decision-making process, aiding in identifying influential variables and potential biases.

2 Background and Related Work

In this section, we will give a brief overview on existing work on automating output checking and human-in-the-loop systems.

2.1 Automating output checking

Output checking is the process of checking the disclosure risk of research results based on microdata files made available in research data centres Bond et al. (2013). Output checking aims to distinguish between output that is safe to be published, output that requires further analysis, and output that is unsafe and cannot be published because of disclosure risk. The output decision is made based on a number of rules. However, research on output checking is still at its early age. Currently, output checking is carried out by human checkers (skilled staff at statistical agencies) which is time consuming, slow, and expensive.
Recently, researchers started looking at different ways to make output checking an easier task. In Domingo-Ferrer and Blanco-Justicia (2021), authors proposed a new approach that leverages machine learning to assist human checkers in output checking. To do so, first, Domingo-Ferrer and Blanco-Justicia (2021) created simulated output checking data based on a subset of rules (called *14 rules of thumb*, see Bond et al. (2013)). Next, they trained a neural network model on each simulated data. Then, they tested how well the rules used to generate log files have been learned and how well the rules that were not used for training have also been captured and learned. Generally, when trying to automate output checking, the main objective is to make a correct decision (safe vs unsafe). From the statistical agency's point of view, we have to minimize false positives. We note that false positives indicate outputs that should not be released (real decision is "unsafe") but whose predicted decision is "safe". In Green et al. (2021), authors proposed a new toolkit *Automatic Checking of Research Outputs* (for short "ACRO"). ACRO is developed in STATA, and extended to Python (and provided as open access on GitHub[2]).

2.2 Guidelines for confidentiality

Guidelines for confidentiality serve as a standard template that should be followed by human checkers and researchers. The guideline is based on rule of thumbs to prevent confidentiality errors and as a result to assess if an output is safe to be released or unsafe. For a detailed and elaborate definitions of rules-of-thumb, we point readers to Bond et al. (2013); Domingo-Ferrer and Blanco-Justicia (2021). In Table 1, we provide a brief description of each rule and corresponding parameters as used in the settings of Domingo-Ferrer and Blanco-Justicia (2021) along with cases in which an output should be treated as unsafe.

2.3 Human-in-the-loop systems

Human-in-the-loop (HITL) approaches (Wu et al., 2022) have applications in a variety of fields, including health care (Wrede and Hellander, 2019), finance, computer-vision (Madono et al., 2020), and natural language processing (Ristoski et al., 2020). HITL is especially relevant in situations where human expertise is necessary for critical decisions, such as medical diagnosis or financial risk assessment. The central idea in HITL is to combine human judgment and intuition with the computational power of algorithms to achieve improved decision outcomes. Depending on who is in control and in which phase of the learning process, we can identify different approaches to HITL in machine learning (Mosqueira-Rey et al., 2023). *Active learning (AL)*, in which the system remains in control of the learning process and humans are involved in the annotation of unlabeled data. *Interactive machine learning (IML)*, in which there is a closer interaction between users and learning systems, with people interactively supplying information in a more incremental way compared to traditional machine learning. *Machine teaching (MT)*, where human domain experts have control over the learning process

[2] https://github.com/eurostat/ACRO

TABLE 1. A summary of rules-of-thumb as used in our experiments and cases in which the output is treated as unsafe.

Type of statistics	Type of output	Decision is "unsafe" if:
Descriptive Statistics	*Frequency tables*	Output is confidential. Some cell contains less than 10 units. A single cell contains more than 90% of the total number of units in a row or column.
	Magnitude tables	Output is confidential. Single cell contains more than 90% of the units in a row or column In some cell the largest contributor contributes more than 50% of cell total.
	Maxima, minima, median, and percentiles	Output is confidential
	Mode	Output is confidential. The frequency of model value is more than 90% of the sample size.
	Means, indices, ratios, indicators	Output is confidential. Sample size <10. A single cell contributes for more than 50% of the total sample.
	Concentration ratios	Output is confidential. Sample size <10. A single cell contributes for more than 90% of the total sample.
	Variance, skewness, kurtosis	Output is confidential. Sample size <10.
	Graph	Output is confidential
Correlation and Regression Analysis	*Linear regression coefficients /* *Non linear regression coefficients*	Output is confidential Intercept is returned as one of the coefficients
	Regression residuals, *Regression residual plots*	Output is confidential
	Test statistics_t *test statistics_F*	Output is confidential Degree of freedom <10
	Correlation	Output is confidential Coefficients that are -1, 0, 1 or <10
	Correspondance analysis	Output is confidential. Number of variables <2 or sample size <10.

by delimiting the knowledge that they intend to transfer to the machine learning model. In addition to the aforementioned benefits of HITL, we also point to the importance of HITL in that it offers a more Explainable AI (XAI), "Usable AI" and "Useful AI" (Mosqueira-Rey et al., 2023; Xu, 2019).

3 Experimental Setup

In this section, we describe our data and the machine learning algorithms that we will use in our experiments.

3.1 Data Set

There is a substantial difference between real log files (i.e., a mix of different output files such as excel files, SPSS files, figures, text) and simulated data. In real world, output checking data present several challenges such as unstructured data, inconsistent column naming, and varying record lengths.
Following Domingo-Ferrer and Blanco-Justicia (2021) and using the same set of rules of thumb and the parameters they used, we generated training data with a total of 200K records and test data with a total of 14000. Every rule has approximately 14700 records in the training data and 1000 records in the test data. For the real test data, we have 125 records mainly dominated by frequency table, magnitude table, and regression model. In our proof-of-concept, we mainly focused on excel files and checking SPSS code to see if it contains sensitive information. Differently from Domingo-Ferrer and Blanco-Justicia (2021), we pre-processed the variables of the simulated data such that we can easily match with real log files. We binarize all variables as follows:

1. Cell units < 10 get 1, otherwise 0,
2. If a single cell contains more than 90% of the total number of units gets 1, otherwise 0 (for group disclosure),

3. If in some cells the largest contributor contributes more than 50% of cell total gets 1, otherwise 0 (for dominance),
4. If intercept is returned the cell gets 1, otherwise 0,
5. If degree of freedom is less than 10 so variable gets 1, otherwise 0.

We note that we excluded from the data the type of analysis (rules). We consider this information as less relevant to the machine learning algorithm, since we challenge the model to capture which rule to use and generate the final decision. Going from real world to simulated data or the other way around aim to tackle the same prediction goal, which is to predict if an output is "safe" (or Decision = True) to be released or "unsafe" (or Decision = False) requiring a second protection (or a modification by researchers).
In Figure 1, we present the distribution of decisions (True or False) based on different analysis types in three data sets: simulated train data, simulated test data, and real test data. Recall that decisions in simulated train data and simulated test data are generated based on rules-of-thumb. Decisions in real test data are created by human checkers when evaluating real log files. Each dataset is represented by a separate sub-figure. The figure shows how the decision variable and analysis types are unequally distributed between simulated data and real data.

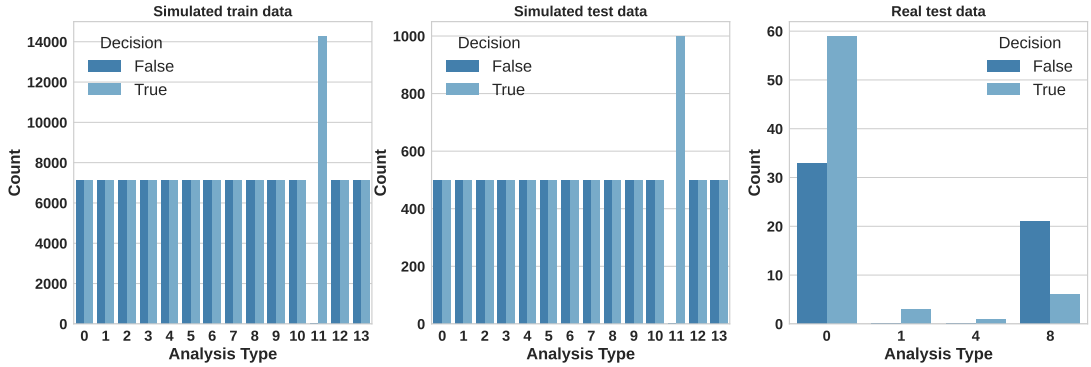

FIGURE 1. Distribution of target variable `Decision` (True= "safe" and False= "unsafe") based on different analysis (rule) types on: (Left): simulated train data, (Middle): simulated test data, (Right): real test data. Analysis type = 11 is about factor analysis and it is set to True (= "Safe") Domingo-Ferrer and Blanco-Justicia (2021).

Figure 2 presents the feature importance plot generated using the LightGBM model. Feature importance provides valuable insights into the contribution of each input variable in making predictions. In Figure 2, the x-axis represents the relative importance score of each variable, while the y-axis displays the variable names. Variables are arranged in descending order of importance, with the most influential variable placed at the top. The importance score reflects the degree of impact each variable has on the model's predictive performance.

3.2 Machine Learning Algorithms

Neural Network The neural network used by Domingo et al. consists of two hidden dense layers, each containing 64 neurons with ReLU activation functions, followed by dropout layers. The networks takes a 12-dimensional input and predicts a binary output using a sigmoid activation function in the final layer (more details about neural network architecture and hyper-parameters tuning can be found in Domingo-Ferrer and Blanco-Justicia (2021)).
LightGBM In addition to the neural network algorithm used in Domingo-Ferrer and Blanco-Justicia (2021), we evaluate the output checking using a lightGBM algorithm (Ke et al., 2017). LightGBM is a gradient-boosting framework based on decision trees to increase the efficiency of the model and reduces memory usage. LightGBM handles different types of Gradient boosting methods which can be specified with the boosting

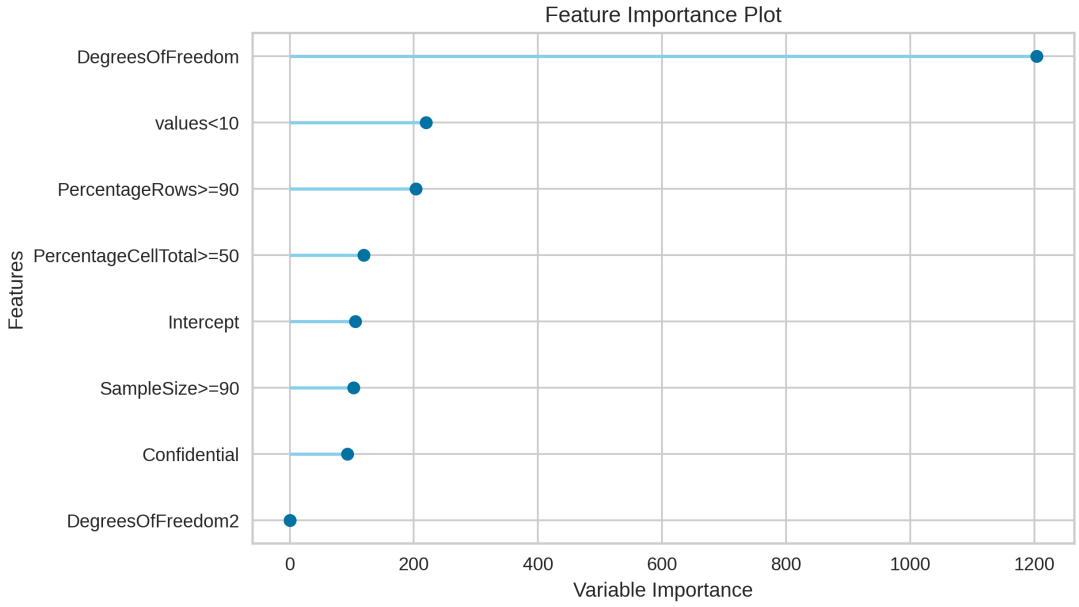

FIGURE 2. Top important features in predicting if an output is safe to be released or not using LightGBM model.

parameter: GBDT (Gradient Boosted Decision Trees), DART (Dropouts meet Multiple Additive Regression Trees), and GOSS (Gradient-based One-Side Sampling). In our experiments, we used the GBDT boosting method. GBDT is a traditional Gradient Boosting Decision Tree and uses several decision trees that are built sequentially. The first tree learns how to fit to the target variable. The second tree learns how to fit to the residual (difference) between the predictions of the first tree and the ground truth. The third tree learns how to fit the residuals of the second tree and so on. We used the implementation of LightGBM in open toolkit PyCaret. We used split based split, learning rate is set to 0.1, minimum number of child samples is set to 20, and $n\_estimator$ is equal to 100.
We compare the performance of the neural network and the lightGBM algorithms to a random classifier. This random classifier serves as a simple baseline to compare against. Our random classifier uses majority class strategy which returns the most frequent class label.

3.3 Evaluation metrics

In order to assess the quality of the target model predictions, we will calculate: confusion matrix, F1 macro-average, Matthews Correlation Coefficient (MCC), and geometric mean (G-mean).
Confusion matrix is a table that is used to define the performance of a classification algorithm in terms of: True Positive (TP), False Positive (FP), True Negative (TN), False negative (FN).
The macro-averaged F1 score (F1-Macro) is computed using the arithmetic mean (aka unweighted mean) of all the per-class F1 scores.[3] This method treats all classes equally regardless of their support values.
geometric mean (G-mean) is the geometric mean of sensitivity and specificity (Sun et al., 2009). G-mean takes all of the TP, TN, FP, and FN into account.

$$\text{G-Mean} = \sqrt{\frac{TP}{TP+FN} * \frac{TN}{TN+FP}} \tag{1}$$

[3]F1 score is the harmonic mean of precision $TP/(TP+FP)$ and recall $TP/(TP+FN)$.

Matthews Correlation Coefficient (MCC) metric also takes into account all of TP, TN, FP, and FN. MCC is a balanced measure that can be used especially if the classes of the target attribute are of different sizes (Chicco and Jurman, 2020). It returns a value between -1 and 1.

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \tag{2}$$

SHAP stands for SHapley Additive exPlanations, is state-of-the-art in Machine Learning explainability. SHAP quantifies how important each input variable is to a model for making predictions (Lundberg and Lee, 2017). This can be a useful sanity check that the model is behaving in a reasonable way. With SHAP [4], we can generate a global interpretation and a local interpretation of a single prediction. The SHAP plots show variables that contribute to pushing the output from the base value (average model output) to the actual predicted value. We use Beeswarm plots to report global interpretability of our machine learning model. A Beeswarm plot is a complex and information-rich display of SHAP values that reveal not just the relative importance of features, but their actual relationships with the predicted outcome. For local interpretability, we use Force plots. A Force plot or reason plot displays key information about an individual case in a more condensed format. For our implementation, we use PyCaret [5], an open-source, low-code machine learning library in Python.

4 Experimental Results

In this section, we provide our results of reproducing state-of-the-art work on using machine learning to assist output checking as described in Domingo-Ferrer and Blanco-Justicia (2021). We extend the results of that paper by testing the trained model on real test data. Then, we involve human-in-the-loop in our predictions and we provide global and local explanations.

4.1 Reproducing Domingo-Ferrer and Blanco-Justicia (2021)

In Table 2, we provide results of prediction performance measured in terms of F1 macroaverage, MCC, G-Mean, TP (true positive), FP (false positive), TN (true negative), FN (false negative). We see that for simulated data and HITL= None, both neural network as used in Domingo-Ferrer and Blanco-Justicia (2021) and our LightGBM (LGBM) outperform the random classifier. We observe that overall neural network classifier performs better than LGBM, except for FP. We note that FP (False Positive) indicates that the model predicts an output as safe but human checker decides that the output is unsafe. We are mainly interested in a low false positive (FP) since FP is dangerous for the privacy of individuals and should thus be avoided as much as possible by statistical agencies. We see that LGBM has an FP = 0 when trained on simulated data. In the remainder of this paper, we will continue our experiments with the LightGBM model.
To further illustrate the performance of the LGBM model when trained and tested on simulated data, we provide the Area Under the Receiver Operating Characteristic (ROC) Curve (AUC-ROC) and the Precision-Recall curve in Figure 3.
On the left side of Figure 3, the AUC-ROC curve is displayed. The ROC curve shows the trade-off between the true positive rate and the false positive rate. We observe that the ROC curve showcases a good prediction ability, with both classes (False and True) achieving an ROC value of 0.91. Also, the micro-average ROC curve, representing the overall performance across all classes, achieves an AUC of 0.93.
On the right side of Figure 3, the Precision-Recall curve is depicted. The Precision-Recall curve evaluates the precision (positive predictive value) against the recall (true positive rate) at various threshold settings. The average precision is calculated by considering the precision-recall trade-off across all thresholds. We see that

[4] https://shap.readthedocs.io/en/latest/index.html
[5] https://pycaret.gitbook.io/docs/

TABLE 2. Prediction performance measured in terms of F1 macroaverage, MCC, G-Mean, TP (true positive), FP (false positive), TN (true negative), FN (false negative). We compare performance of random classifier to LGBM (LightGBM) and neural network. Our classifiers are trained on simulated data. We evaluate trained models on simulated test data and real test data. Random classifier uses majority class strategy. HITL stands for human-in-the-loop.

Data Sets	*HITL*	Classifier	**F1 (Macro)**	**MCC**	**G-Mean**	**TP**	**FP**	**TN**	**FN**
Simulated Data	*None*	*Random*	0.3488	0.0000	0.5000	0	6500	0	7500
		LGBM	0.8489	0.7376	0.8599	6500	0	2101	5399
		Neural Network	0.9421	0.8838	0.9421	6123	377	433	7067
Real test data	*None*	*LGBM*	0.6139	0.4052	0.6409	16	**38**	1	68

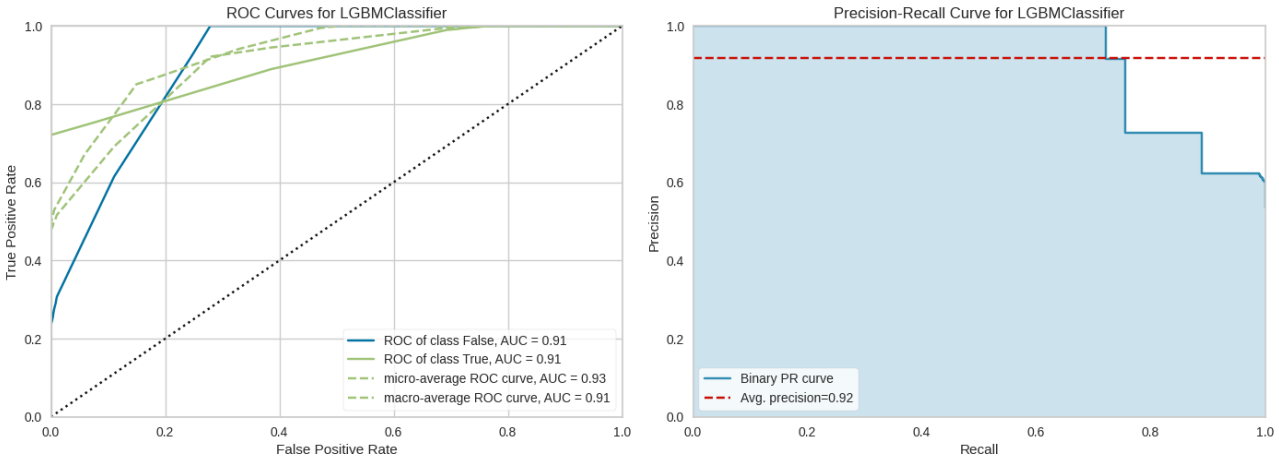

FIGURE 3. Performance of LightGBM (LGBM) model when trained and tested on simulated data: (Left) AUC-ROC curve, (Right) Precision-Recall curve.

our LightGBM (LGBM) model achieves an average precision score of 0.92, highlighting its ability to maintain high precision even when recall is taken into account.

4.2 Evaluation of LGBM trained model on real test data

Now, we move to test our LGBM trained model on test real data. Our aim is to see whether a model trained on simulated data could generalize and maintain good prediction performance when tested on real data. Results are provided in Table 2 (Real test data with HITL= None). We observe a difference in performance between the LightGBM model when tested on simulated test data against the same LightGBM model when tested on real data. This result is expected and can be explained by the fact that the real test data has different characteristics than the simulated test data as was shown in Figure 1.

4.3 Incorporating Human-in-the-Loop with COACH

In this section, we will describe our process of incorporating a human in the automation of output checking. We have developed an interactive app, called "COmputer-Assisted output CHecking with Human-in-the-loop", `COACH` for short. COACH serves as a critical interface between our LightGBM model and human checker experts. COACH allows human users to assess the LightGBM model's predictions. It provides an efficient and transparent means for human input, ensuring that human expertise is leveraged in complex decision scenarios. Figure 4 shows an example of human-machine interaction.

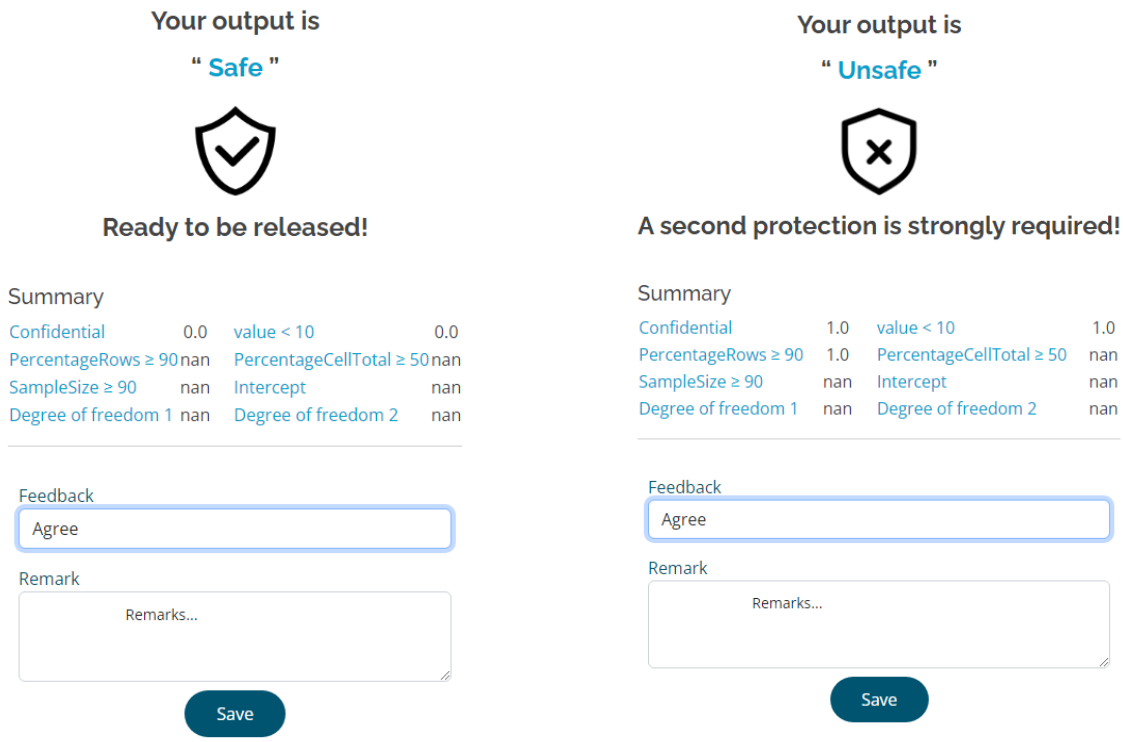

FIGURE 4. An example of Human-Machine interaction. The model returns a prediction = `Unsafe`. Human checker agrees with the ML prediction because there are values less than 10 and a possible group disclosure.

As depicted in Figure 4, we incorporated a `feedback` box that offers two primary options: "Agree" and "Disagree". When presented with a decision, the human checker can choose to "Agree" if they concur with the model's prediction. Alternatively, if the human checker disagrees, they can select "Disagree" and are prompted to provide an elaborate text explaining the reasons behind their decision.
For records (or queries) where the human checker disagrees with the model's prediction, we view it as a valuable learning opportunity. The elaborated text provided by the human checker offers crucial insights into potential areas where the model may have limitations or biases. This feedback becomes an essential part of our training data and helps us to identify and rectify our LightGBM model shortcomings. After incorporating human feedback into the training data, we undertake data augmentation, enriching our data set with diverse scenarios and real-world insights. Subsequently, we retrain our LightGBM model using this updated data set, empowering it to learn from the collective knowledge and expertise of both human experts and historical data.

TABLE 3. Prediction performance measured in terms of F1 macroaverage, MCC, G-Mean, TP (true positive), FP (false positive), TN (true negative), FN (false negative). We compare performance of random classifier to LGBM (LightGBM) and neural network. Our classifiers are trained on simulated data. We evaluate trained models on simulated test data and real test data. Random classifier uses majority class strategy. HITL stands for human-in-the-loop.

Data Sets	*HITL*	Classifier	**F1 (Macro)**	**MCC**	**G-Mean**	**TP**	**FP**	**TN**	**FN**
Simulated Data	*With*	*Random*	0.3488	0.0000	0.5000	0	6500	0	7500
		LGBM	0.8489	0.7376	0.8599	6500	0	2101	5399
Real test data	*With*	*LGBM*	0.9099	0.8229	0.9143	51	**3**	8	61

Our results of incorporating human-in-the-loop are provided in Table 3. Comparing the performance of lightGBM on real test data before adding human-in-the-loop in Table 2 and after adding human-in-the-loop in Table 3, we see that the latter helped to improve the prediction performance. Specifically, we point to the positive impact of human-in-the-loop in reducing false positive (FP) cases (from 38 to 3). This confirms that retraining based on human feedback enables our model to adapt to dynamic and evolving scenarios of the real world, which result in an improvement in performance over time. In addition, human feedback helps us to uncover potential biases in the model's predictions, allowing us to take proactive steps to address and rectify such biases. Also, with human-in-the-loop we envision the trained model to be able to easily adjust and adapt to new rules.

4.4 Utilizing Global and Local SHAP Values

In this section, we look at understanding how our variables contribute to the predictions made by LightGBM model. Todo so, we use SHAP since it helps in model interpretability and provides valuable insights into the underlying decision-making process of the model. In Figure 5, we provide results of global interpretability of our LightGBM model using the collective SHAP values before and after involving human-in-the-loop. Figure 5 provides valuable insights into how individual variables contribute to the predictions made by the LightGBM model. We observe that high values of the "Confidential" variable have a high negative contribution on the prediction, while low values have a high positive contribution. In Figure 5, we observe that the "Confidential" and "values<10" variables have a low positive contributions. The variables "Intercept" and "DegreesOfFreedom2" have almost no contribution to the prediction, whether its values are high or low.

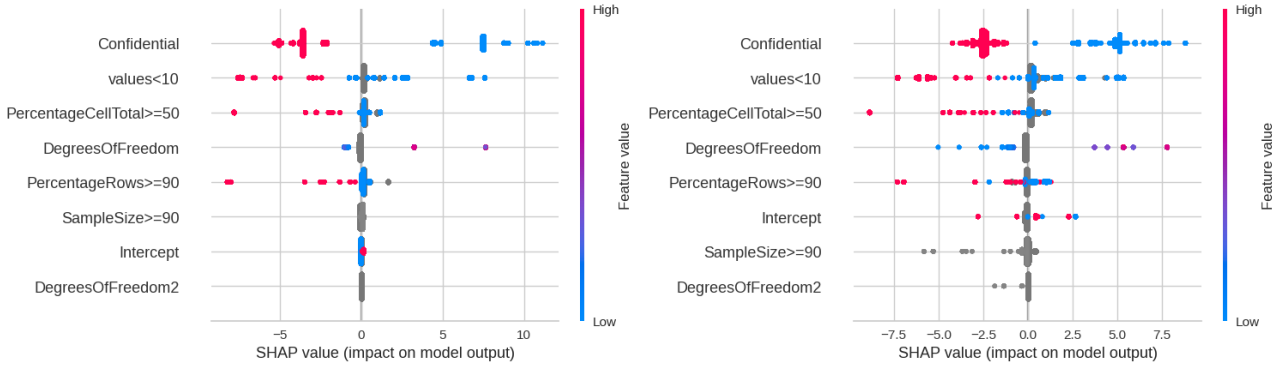

FIGURE 5. Global interpretability: the collective SHAP values show how much each predictor contributes, either positively or negatively, to the target variable. (Left) shows SHAP summary from simulated train data, (Right) shows SHAP summary from simulated train data after involving human-in-the-loop. If the value of a variable for a particular record is relatively high, it appears as a red dot, and relatively low variable values appear as blue dots.

Figure 6 shows an example of local interpretation for one observation from test data. We show the difference in reasoning plot before (up) and after (bottom) incorporating human-in-the-loop. Feature values causing increased predictions (towards decision= "unsafe") are in pink, and their visual size shows the magnitude of the feature's effect. Feature values decreasing the predictions are in blue. We see that the biggest impact comes from variables "Confidential" = 1 and "values<10" = 1 (there are values below 10 that could leak sensitive information). Though the "percentageCellTotal>=90" = NaN value has a slight effect decreasing the prediction.

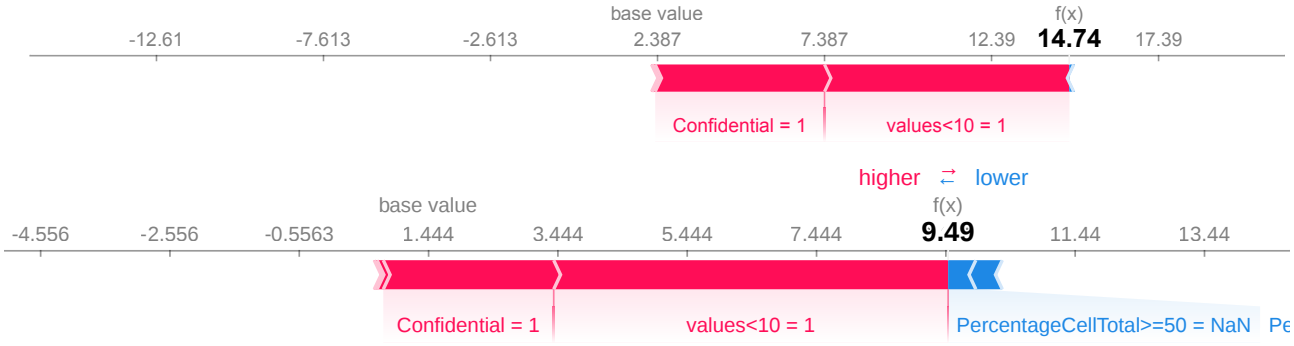

FIGURE 6. Local interpretability using reasoning plots for an individual case in test data. Variables with SHAP values that "push" the model towards `Unsafe` decision appear on the left in red, whereas Variables with SHAP values that "push" the model towards `safe` decision appear on the right in blue. (Up) Reasoning plot before involvement of human-in-the-loop. (Bottom) Reasoning plot after involvement of human-in-the-loop.

5 Conclusion and Future Work

In this paper, we propose COACH, a novel approach to semi-automate output checking by incorporating human-in-the-loop interactions. Our contributions include extending existing work on automating output-checking algorithms, creating the COACH App to involve human checkers, and utilizing global and local SHAP values for model explainability. The integration of human judgment enriches the output-checking process, leading to more accurate, transparent, and trustworthy output predictions. Through COACH, we aim to bridge the gap between fully automated output checking and the need for human oversight, advancing state-of-the-art output validation methodologies.

In the future, we plan to extend COACH's application to other statistical offices, ensuring its success across different institutes. We also aim to enhance COACH by exploring other pre-processing strategies that better reflect real-world data complexities, reducing bias and improving data representation.

Acknowledgment

We would like to thank `Microdata Services` at Centraal Bureau voor Statistiek, the Netherlands for providing us with data and knowledge on output checking. Also, we thank Malek Slokom for her guidance on creating COACH App.

References

Bond, S., M. Brandt, and P.-P. de Wolf (2013). Data without boundaries: Standalone document guidelines for output checking. [Online; accessed in 10 October 2022].

Chicco, D. and G. Jurman (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics 21*(1), 1–13.

Domingo-Ferrer, J. and A. Blanco-Justicia (2021). Towards machine learning-assisted output checking for statistical disclosure control. In *Proceedings of 18th International Conference on Modeling Decisions for Artificial Intelligence: MDAI 2021*, Berlin, Heidelberg, pp. 335â345. Springer-Verlag.

Green, E., F. Ritchie, and J. Smith (2021). Automatic checking of research outputs (acro): A tool for dynamic disclosure checks. *ESS Statistical Working Papers 2021 Edition*.

Ke, G., Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pp. 3149–3157.

Lundberg, S. M. and S.-I. Lee (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems 30*, 4765–4774.

Madono, K., T. Nakano, T. Kobayashi, and T. Ogawa (2020). Efficient human-in-the-loop object detection using bi-directional deep sort and annotation-free segment identification. pp. 1226 â 1233.

Mosqueira-Rey, E., E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, and Á. Fernández-Leal (2023). Human-in-the-loop machine learning: A state of the art. *Artificial Intelligence Review 56*(4), 3005–3054.

Ristoski, P., A. L. Gentile, A. Alba, D. Gruhl, and S. Welch (2020). Large-scale relation extraction from web documents and knowledge graphs with human-in-the-loop. *Journal of Web Semantics 60*, 100546.

Sun, Y., A. K. Wong, and M. S. Kamel (2009). Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence 23*(04), 687–719.

Wrede, F. and A. Hellander (2019). Smart computational exploration of stochastic gene regulatory network models using human-in-the-loop semi-supervised learning. *Bioinformatics 35*(24), 5199–5206.

Wu, X., L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He (2022). A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems 135*, 364–381.

Xu, W. (2019, jun). Toward human-centered ai: A perspective from human-computer interaction. *Interactions 26*(4), 42â46.