A Framework for Building Uncertainty Wrappers for AI/ML-based Data-Driven Components

Michael Kläs ✉ and Lisa Jöckel ✉

Fraunhofer Institute for Experimental Software Engineering IESE,
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
{michael.klaes, lisa.joeckel}@iese.fraunhofer.de

Abstract. More and more software-intensive systems include components that are data-driven in the sense that they use models based on artificial intelligence (AI) or machine learning (ML). Since the outcomes of such models cannot be assumed to always be correct, related uncertainties must be understood and taken into account when decisions are made using these outcomes. This applies, in particular, if such decisions affect the safety of the system. To date, however, hardly any AI-/ML-based model provides dependable estimates of the uncertainty remaining in its outcomes. In order to address this limitation, we present a framework for encapsulating existing models applied in data-driven components with an uncertainty wrapper in order to enrich the model outcome with a situation-aware and dependable uncertainty statement. The presented framework is founded on existing work on the concept and mathematical foundation of uncertainty wrappers. The application of the framework is illustrated using pedestrian detection as an example, which is a particularly safety-critical feature in the context of autonomous driving. The Brier score and its components are used to investigate how the key aspects of the framework (scoping, clustering, calibration, and confidence limits) can influence the quality of uncertainty estimates.

Keywords: Artificial Intelligence, Machine Learning, Safety Engineering, Data Quality, Operational Design Domain, Out-of-Distribution, Dependability

1 Introduction

Components based on machine learning and other AI methods are increasingly finding their way into software-intensive systems. In this context, we talk about data-driven components (DDCs) if the functionality provided by the component is not explicitly specified and implemented by developers, but is automatically generated by algorithms based on data. Such data-driven components play an important role, particularly in areas such as autonomous driving or Industry 4.0, because they provide opportunities for perception that cannot yet be reasonably realized with conventional software. Well-known tasks are the recognition of people, traffic signs, or other objects and structures in camera images, but also speech recognition and natural language processing.

Especially when applied in the context of safety-critical systems, there is, however, the challenge that – in contrast to traditional software – we can neither assume nor demonstrate that data-driven components will provide the intended output for any input. For example, we cannot assume that a person recognition component will really recognize all types of people in any operating condition. The outputs of data-driven components are thus always subject to uncertainty. Therefore, it seems essential to quantify the degree of uncertainty, which is usually also situation-dependent, and consider it in subsequent decision-making. For example, when the outcome of a pedestrian being detected at a distance of 40 meters has a moderate uncertainty, this could lead to a precautionary reduction in speed, whereas when the outcome is highly uncertain, i.e., if there is probably no pedestrian, the current speed may be maintained as long as sufficient time remains for a braking maneuver at short notice.

Existing data-driven models (DDMs) and modeling approaches, however, usually do not explicitly consider uncertainty in their outcomes, or provide uncertainty estimates that are not dependable from a statistical point of view [1]. To address this limitation, which we discuss further in the background section of this paper, Kläs and Sembach introduced the mathematical foundations for '*uncertainty wrappers*', which enclose an existing data-driven model and enrich its outcomes with dependable uncertainty estimates [2]. The concept considers the three different kinds of common sources of uncertainty introduced in the onion shell model: limitations regarding the *model fit*, *data quality*, and *scope compliance* [3].

Contribution: To make uncertainty wrappers applicable in practice, we developed a framework, which we present in this paper. The framework underpins the previously published mathematical concepts with specific methods and provides a reference architecture as well as tooling support for building and applying uncertainty wrappers. In order to show its applicability and potential benefits, we illustrate the application on an existing DDM for pedestrian detection and investigate how the key elements of the framework (scoping, clustering, calibration, and confidence limits) influence the quality of uncertainty estimates as measured by the Brier score and its components.

Structure: Section 2 motivates the relevance of the proposed framework, positioning it in the context of related work and introduces the Brier score measure. Section 3 presents the framework and illustrates its use on a simplified example of pedestrian detection. Section 4 investigates which parts of estimation quality as measured by the Brier score are addressed by the key aspects of the framework. Finally, Section 5 concludes the paper with previous application experience and an outlook on future work.

2 Background

In this section, we will first provide a short summary on uncertainty estimates and observed limitations in their calculations (see [1] for a more elaborate discussion). Next, we will briefly introduce the foundations of the uncertainty wrapper concept that we use and elaborate in this paper. Finally, we will introduce the Brier score as a measure for evaluating uncertainty estimates.

Uncertainty Estimates: Uncertainty estimates for categorical outcomes, which are commonly provided as probabilities, can be obtained by different means. There are, for example, specific kinds of DDMs that implicitly provide uncertainty estimates, such as decision trees, which provide not only the selected category, but also its probability. Several ML approaches have also been extended to provide DDMs with uncertainty estimates (e.g., various Neural Networks [4] [5]). For example, Henne et al. [6] and Snoek, et al. [7] provide benchmarking for a selection of such approaches. However, uncertainty estimates directly provided by DDMs are usually not dependable, i.e., there is no statistical guarantee for these values. Sometimes the provided values are not even probabilities in a probabilistic sense, such as in the case of Naïve Bayes and Support Vector Machines. Moreover, DDMs usually focus on providing accurate prediction results, not uncertainty estimates. Therefore, they ignore uncertainty-relevant features that do not contribute to the accuracy of the model. Finally, their estimates are usually calculated based on data used during model training, which increases the risk of overfitting and thus of overconfident estimates [1].

There are approaches that can be applied to calibrate improper probability estimates using an independent representative calibration dataset [8]. Scikit-learn, e.g., provides algorithms for *sigmoid* (parametric) and *isotonic* (nonparametric) calibration [9]. However, the use of calibration methods does not solve the problem that the provided uncertainty estimates are usually received from a black box; e.g., domain experts cannot semantically validate the criteria based on which the model decides whether a certain result has a higher or lower attributed uncertainty. Moreover, the calibration methods of which we are aware provide no upper boundary for the estimated uncertainty given a requested level of statistical confidence, which limits their usefulness in a safety argument. Finally, existing approaches largely ignore the fact that a DDM might also be applied outside the scope for which it was calibrated and that the provision of realistic uncertainty estimates and of accurate outcomes does not necessarily require the same inputs and features, which is, however, an implicit assumption when integrating the calculation of uncertainty estimates directly into a DDM [1].

Uncertainty Wrapper: The model-agnostic concept of an uncertainty wrapper proposed by Kläs & Sembach [2] addresses these limitations. It defines uncertainty as the likelihood that the outcome of a DDM is not correct considering a given definition of correctness. Based on this definition, Kläs & Sembach show that uncertainty can be mathematically decomposed into the three classes of an onion shell model [3]: (1) Model-fit-related uncertainty occurs due to the inherent modeling limitations when creating a DDM. (2) Quality-related uncertainty results from applying a DDM on input data with quality limitations, which is a common phenomenon in practice. (3) Finally, scope-compliance-related uncertainty addresses circumstances where a DDM is applied for cases outside the application scope for which it was built and tested.

Where model-fit-related uncertainty can be determined with traditional model testing approaches, the likelihood of scope compliance is determined by '*scoping*'. Scoping checks the adherence of a specific case to a number of scope factors that should all be valid for each case within the intended application scope. Quality-related uncertainty is addressed by *clustering* the cases in the application scope into areas with similar uncertainty considering relevant quality factors. The estimates for individual clusters need to

be *calibrated* on a dataset representative for the intended application scope. Finally, considering a requested level of *confidence*, statistics are proposed for estimating an upper boundary for the uncertainty in each cluster.

Although the concept of uncertainty wrappers appears promising, a framework operationalizing the concept has been missing to date, along with a reference architecture for the wrapper, specific methods that can be applied for clustering and scoping, as well as tooling support.

The work most closely related to the framework proposed in this paper may be the framework proposed by Czarnecki and Salay for managing perceptual uncertainty [10], which, however, remains on a more descriptive level. Whereas the uncertainty wrapper focuses on situation-aware estimates at runtime, known tooling support usually focuses on dealing with uncertainty at design time. For example, Matsuno et al. [11] provide a tool that investigates how uncertainty in ML affects safety arguments, and the *nn-dependability-kit* examines the robustness of data-driven components against known perturbations during testing [12].

Evaluating Uncertainty Estimates: Defining uncertainty as the probability that the DDM outcome is not correct, we can consider uncertainty estimation as a (binary) probabilistic classification task. With scoring rules, decision theory provides a means for evaluating the utility of such estimates. We generally request certain properties for scoring rules to be reasonable; the most relevant one is that they should be strictly proper. Strictly proper scoring rules assure that the scoring result only depends on the probability to be estimated and is optimized exclusively by estimating the correct probability.

Theoretically, the number of strictly proper scoring rules is infinite. However, there are some popular rules such as negative log-likelihood and especially ***Brier score (bs)***, which was introduced by Brier in 1950 [13] and is, e.g., applied by Snoek, et al. [7]. It measures the mean squared difference between the predicted probability of an outcome and the actual outcome. Applied to uncertainty, this means that if the estimated uncertainty is 80% and the actual outcome is wrong (e.g. '1'), $bs = (1\text{-}0.8)^2 = 0.04$. So it can be considered as a cost function with a minimum of 0, where lower values mean more accurate uncertainty estimates. Choosing the Brier score as a cost function is especially appealing if we do not know how the estimates are planned to be used further, since it does not emphasize particular decision thresholds but assumes a uniform distribution.

Murphy was able to show that the Brier score can be further detailed by decomposing it into three additive components, which he named uncertainty, resolution, and reliability [14]. Deviating from this designation, we speak of variance instead of uncertainty to avoid confusion, and of unreliability instead of reliability since counterintuitively, low reliability values would mean high reliability. Besides Brier score, we consider its three components since they provide a more detailed picture of the specific limitations affecting uncertainty estimates, which are also addressed by different elements of the uncertainty wrapper framework:

$$bs = var - res + unr \tag{1}$$

Variance (var) describes the empirically observed variation in the correctness of the DDM outcomes (i.e., Bernoulli variance). This means DDMs with a high average error rate provide a high average uncertainty and thus a high variance.

$$var = E\big(P(correct)\big)\, E\big(P(wrong)\big) \tag{2}$$

Resolution (res) describes how much the case-specific uncertainty estimates differ from the empirically observed average error rate of the DDM (i.e., the case-independent average uncertainty).

$$res = E\left(P(wrong \mid uncertainy) - E\big(P(wrong)\big)\right)^2 \tag{3}$$

Unreliability (unr) describes how much, given an estimated uncertainty, the empirically observed uncertainty (i.e., the error rate) differs. If we have, for example, ten cases with an estimated uncertainty of 30%, *unr* is 0 if we empirically observe that the DDM outcome for three of them is wrong.

$$unr = E\big(uncertainy - P(wrong \mid uncertainty)\big)^2 \tag{4}$$

3 Framework and Application Example

This section introduces a framework for building uncertainty wrappers for arbitrary data-driven models, assuming the availability of a labeled dataset that is representative of the model's *target application scope* (TAS), i.e., its intended application settings. TAS as a concept is thus comparable to the operational design domain as defined by SAE J3016 for automated driving. A dataset or sample is considered as *representative* if it "ensures external validity in relationship to the population of interest the sample is meant to represent" [15], e.g., by using a random selection approach. Besides the *input* of the data-driven component, the dataset also needs to comprise the *intended outcome* for each case (e.g., the location of a pedestrian in a given input image). Given a definition of *correctness*, we can apply the DDM for each case in the dataset and derive whether the outcome is correct or not.

Fig. 1 illustrates how the uncertainty wrapper architecture complements the DDM with a *quality model* and a *quality impact model* to determine quality-related uncertainties and a *scope model* and a *scope compliance model* to determine the likelihood of scope incompliance. Finally, the wrapper combines both results into an overall uncertainty statement considering the requested level of *confidence*.

In this setting, *uncertainty* is defined by the likelihood that outcomes of the DDM are not correct, and a *dependable uncertainty estimate* is a justified upper boundary on this for a given level of confidence [2]. In our application, we get as an outcome a bounding box with, e.g., the corners (34, 352) and (51, 359) that could potentially contain a pedestrian, extended with the dependable uncertainty estimate that the probability of providing a wrong box is less than 4%, considering a confidence level of .9999 and the definition of correctness.

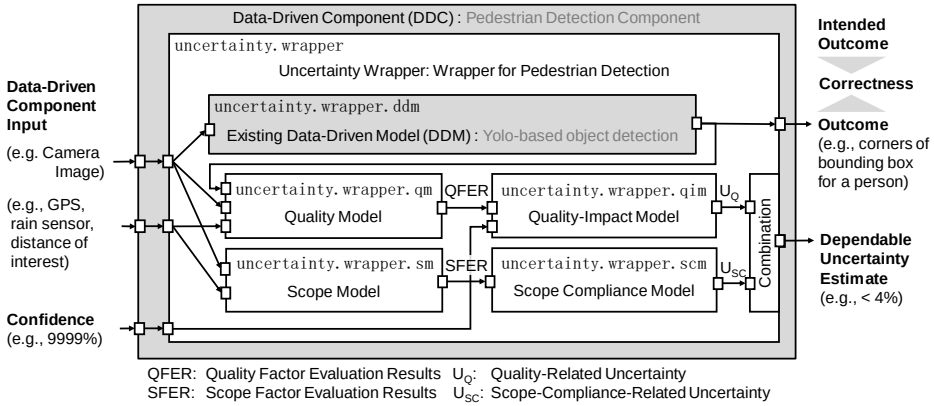

Fig. 1 Wrapper architecture including the dataflow between the data-processing models

The tooling that is part of the framework supports the creation and validation of each uncertainty wrapper element in a separate module developed in Python, the de-facto standard language for data science in most companies (cf. [16]). The elements realizing the overall wrapper as well as elements provided in submodules have been validated to be compliant with the estimator interface of the popular open-source machine learning package 'scikit-learn' [9]. As a result, the elements are intuitive to use for most data scientists, can simply reuse existing models and metrics from scikit-learn, and can be easily integrated into larger data analysis pipelines or ensembles of models.

The sections below illustrate each element using a simplified example application of pedestrian detection on a dataset containing approx. one million images. Although the framework was also applied and validated in an industry setting with field data, we decided to illustrate its usage on YOLOv3 as a publicly available DDM [17] and with data generated using the driving simulator CARLA [18], which allows us to publish concrete numbers as well as raw data (upon request). The given example primarily serves to illustrate the framework; it does not claim high external validity.

Scope Models contain all elements required to process the inputs of a DDC in order to provide case-specific information on 'scope compliance'-related causes for uncertainty considering a set of scope factors. Each scope factor is quantified by one or more scope measures, which provide measurement results that are then evaluated with a scope factor evaluation. For example, if the TAS of the DDC is limited to Germany, we need to find out whether the model is applied outside Germany, since the data we used to test the model and calibrate the uncertainty estimates might not be representative for usage outside of Germany. Therefore, we would need to define measures that extract the geolocation from the DDC input and a factor that applies an evaluation returning the likelihood that the geolocation is within Germany.

```
sm = ScopeModel(List[ScopeFactor(List[ScopeMeasure], ScopeFactorEval)]).fit(...)
scope_factor_eval_results = sm.predict(ddc_input)
```

The framework mainly distinguishes three kinds of scope factors (Fig. 2). *Boundary-based* factors define a valid range for specific DDC inputs. These boundaries can be explicitly stated in the TAS definition, such as requesting a geolocation within Germany, or implicitly derived from empirical data, such as a valid range for temperatures. *Condition-based* factors model multivariate concepts like location-specific temperature ranges. Finally, *novelty-detecting factors* try to detect cases that are outside the TAS but still satisfy boundary- and condition-based scope factors. This is important since we would usually need an infinite number of scope factors to describe a TAS perfectly.

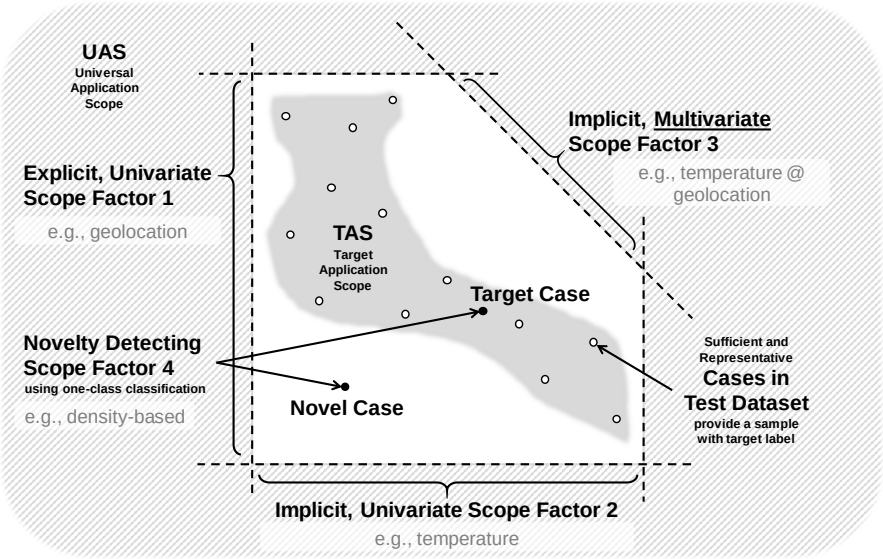

Fig. 2 The different kinds of scope factors help to detect a potential application outside the TAS.

In order to detect such novel cases, the framework relies on *one-class classification*, considering the cases of the test dataset as representations taken from the TAS. The framework supports the creation of novelty-detecting factors based on one-class support vector machines, kernel-density estimation, and percentile-based boundaries [19]. Moreover, the framework calculates performance statistics for the novelty-detecting factors using cross-validation. The *false alarm rate* is calculated by the ratio of cases that are predicted as novel in hold-out parts of the test data. Since, as for any one-class classification task, we do not have a representative set of cases labeled as novel, the framework supports sampling cases from the input space defined through boundary- and condition-based scope factors (i.e., the white area in Fig. 2) assuming a given, e.g., equal, distribution. The obtained *overall alarm rate* (incl. correct as well as false alarms, which cannot be distinguished due to missing labels) can then be related to the false alarm rate in order to select the most appropriate novelty-detecting factor.

Scope Compliance Models provide for a given case an estimate of the uncertainty jointly introduced by 'scope compliance'-related causes considering the factor-individual evaluation results provided by the scope model. The implementation applies the

multiplicative combination proposed in [2] and finally returns an estimate for the likelihood that the model will be used outside its intended application scope TAS.

```
scm = ScopeComplianceModel(ScopeModel,MultiplicativeCombinationModel(epsilon)).fit(...)
scope_rel_uncert = scm.predict(scope_factor_eval_results)
```

Quality Models contain all elements required to process the inputs of a given DDC in order to provide case-specific information on 'data quality'-related causes of uncertainty. A quality model thus represents the counterpart to the scope model and follows the same structure. Quality factors can be identified by domain experts, but also through data analysis. For example, a low sun altitude may influence the performance of the DDM used for pedestrian detection. Unlike the use of the DDM outside Germany, the use during low sun altitude may still be part of the TAS but makes the detection task harder due to backlight. Therefore, measures can be defined that extract the sun's location and the driving direction from the DDC input. The measurement results are then evaluated using factor evaluation to determine the binary evaluation result low_sun_altitude. Further examples of factors are the amount of precipitation based on the rain sensor signal (optionally complimented with a convolution neural network trained to detect rain) and the distance over which we want to detect pedestrians (which may depend, among other things, on the current vehicle speed).

```
qm = QualityModel(List[QualityFactor(List[QualityMeasure], QualityFactorEval)]).fit(...)
quality_factor_eval_results = qm.predict(ddc_input)
```

Quality Impact Models provide for a given case an estimate of the uncertainty jointly introduced by 'data quality'-related causes considering the factor-individual evaluation results provided by the quality model. In order to make the resulting uncertainty estimate not only statistically sound but understandable and traceable for safety engineers and domain experts, we train an information-gain-based decision tree structure to identify clusters with similar uncertainties (cf. Fig. 3). The dependent variable is the correctness of the DDM outcome and the independent variables are the quality factor evaluation results of the quality model. In order to provide statistically sound results, the clusters (i.e., the tree nodes) are identified using a training dataset and then calibrated with a separate calibration dataset representative of the TAS. Moreover, the given confidence level is considered when the uncertainty estimates are calculated.

```
qim = QualityImpactModel(QualityModel,DecisionTreeAlg,confidence).fit(...).calibrate(...)
quality_rel_uncert = qim.predict(quality_factor_eval_results)
```

The results can be evaluated semantically based on the decision tree structure; e.g., the distance to the pedestrian increases the uncertainty. Moreover, the appropriateness of the refinement can be evaluated comparing the base rate (dashed line in Fig. 3) against the degree of separation provided by the identified clusters measured as loss of certainty (cl). Given a disjoint test dataset, the Brier score results can also be considered.

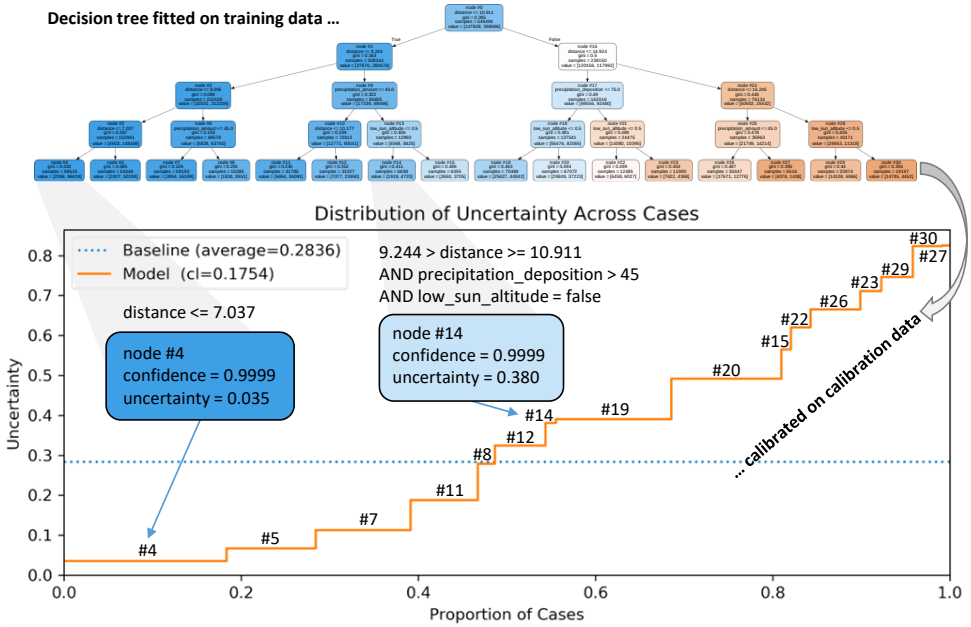

Fig. 3 A calibrated decision-tree-based quality impact model with confidence=.9999 and its evaluation, visualizing uncertainty estimates and certainty loss (cl) in comparison to DDM baseline.

Finally, the calibration can be checked using a calibration curve [20]. The calibration curves in Fig. 4 show that applying a confidence level during calibration pushes the calibration line above the perfect line, making the uncertainty estimates more conservative but also more reliable when tested on new datasets not used for calibration. For example, the uncertainty of the cases in cluster #14 is no longer underestimated.

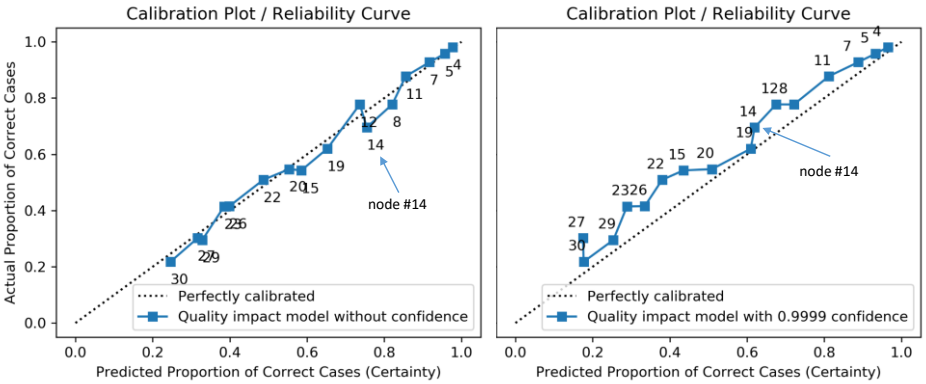

Fig. 4 The calibration curves illustrate the differences between a quality impact model calibrated without a confidence level (left) and one calibrated with a confidence level of .9999 (right).

Wrappers provide a transparent option for enriching existing DDMs with uncertainty estimates, taking the inputs of the DDC and providing uncertainty estimates with a requested level of confidence as a second output. Internally, they delegate the task to the encapsulated DDM and the introduced wrapper elements.

```
wrapper = Wrapper(DataDrivenModel, QualityImpactModel, ScopeComplianceModel)
ddm_outcome, uncertainty = wrapper.predict(ddm_input, confidence)
```

4 Brier-Score-based Investigation

In order to investigate the influence of the key aspects of an uncertainty wrapper – *scoping*, *clustering*, *calibration*, and *confidence limits* – on the quality of the provided uncertainty estimates, we proceeded as follow:

(1) We prepared three separate datasets using the driving simulator CARLA. The *training dataset* was used to identify clusters based on the quality impact model and provide naïve baseline estimates based on the average error rate. The *calibration dataset* representing the TAS was used to calibrate the quality impact model. The *test dataset* was used to evaluate the uncertainty estimates by calculating the Brier score and its components. In order to simulate common issues observed in practice, we modified the distribution of the different kinds of pedestrians between the training and the calibration dataset (e.g., more children). We also kept some unintended cases from the simulation results that were not compliant with our TAS definition in the test dataset, such as pedestrians detected beyond the maximum intended application distance.

(2) Next, we used the framework to create five (partial) wrapper instances A to E:

A addresses *none* of the key aspects. The provided uncertainty estimates are case-independent using the average error rate determined on the training dataset.

B addresses *scoping*, which means that it uses a scope compliance model (cf. Fig. 3) to detect cases that are outside the TAS and rejects them. Estimates are still case-independent using the average error rate determined on the training dataset.

C addresses not only scoping, but also *clustering*, meaning it uses a quality impact model based on a decision tree, which was learned on the training dataset using information gain and a maximum depth of 4. Estimates are now case-aware considering quality factors such as the amount of precipitation measured by the rain sensor to determine the appropriate cluster (i.e., leaf in the decision tree).

D addresses not only scoping and clustering, but also *calibration*, meaning the uncertainty-related values in the leaves of the decision tree (i.e., our clusters) are updated based on the calibration dataset.

E finally addresses all key aspects, which adds *confidence limits*. This means for our example that an upper boundary for the uncertainty estimate is determined for each cluster. The calculations consider not only the specific uncertainty, but also the requested confidence level of .9999 and the number of cases in the cluster.

(3) Finally we applied the wrapper instances A to E to our test dataset with 291,584 data points. On the results, we calculated the Brier score and its components, which we

introduced in Section 2. Moreover, the proportion of clusters providing overconfident uncertainty estimates (o/confident) were determined as a measure of dependability. The uncertainty estimates of a cluster are considered as overconfident if the observed rate of wrong DDM outcomes exceeds the cluster-adjusted uncertainty estimate.

Table 1 summarizes the investigation results. Metrics that change from the preceding wrapper instance to the current one are printed in bold. At this point, we would like to point out once again that the reported results were obtained for data based on images generated by a driving simulator and the application of a general-purpose object detection DDM. Thus, the absolute numbers should be interpreted with care. However, the general effects and tendencies we report are confirmed by our experience when we applied the framework on real-world datasets in an industry setting.

Table 1. Characteristics and performance of several (partial) wrapper instances

	Instance A	Instance B	Instance C	Instance D	Instance E
Scoping	-	✓	✓	✓	✓
Clustering	-	-	✓	✓	✓
Calibration	-	-	-	✓	✓
Confidence	-	-	-	-	✓
#outcomes	291584	242576*	242576*	242576*	242576*
#clusters	1	1	16	16	16
Brier Score	0.28339	0.23995	0.18319	0.15485	0.15502
+ Variance	0.23924	**0.22009**	0.22009	0.22009	0.22009
– Resolution	0.00000	0.00000	**0.06526**	0.06526	0.06526
+ Unreliability	0.04416	**0.01986**	**0.02836**	**0.00002**	**0.00019**
o/confident	100%	100%	100%	**56%**	**0%**

*Further cases (cf. Instance A) were identified by the wrapper as being outside the TAS

Discussion: Instances without clustering have only one cluster comprising all cases, whereas instances C-E are based on the same 16 clusters identified during clustering. Scoping, which detects and indicates situations for which the DDC was not intended, is considered in instances B-E, which is reflected by the reduced number of valid DDM outcomes. Scoping influences variance and unreliability (cf. Instance A vs. B). Variance is reduced since applications outside the TAS are more likely to lead to higher error rates, and unreliability is reduced since situations outside the TAS are usually not well represented in the training data. Clustering increases the resolution by providing uncertainty estimates based on the 16 different clusters identified (cf. Instance B vs. C). The separation into individual clusters, however, can also increase overall unreliability to some extent. Calibration then addresses unreliability, with a representative calibration dataset able to reduce unreliability even to close to zero (cf. Instance D). Usually, ~ 50% of the clusters will now provide overconfident estimates. Depending on the chosen confidence level, the ratio of clusters with overconfident estimates can be reduced to zero or at least to close to zero (cf. Instance E). Depending on the number of clusters

and calibration data points, this can, however, increase unreliability, since the estimate now has to consider some 'safety' margin. In total, brier score is reduced from 0.28339 to 0.15502 by the uncertainty wrapper.

5 Conclusion

The uncertainty wrapper framework differs from existing solutions for dealing with uncertainty in AI/ML-based components. It makes the sources contributing to uncertainty transparent, enabling them to be evaluated by experts instead of being hidden in the algorithms of the DDM or its extensions (cf., e.g., approaches benchmarked in [6]). Moreover, unlike in existing solutions, a requested confidence level, which can even change at runtime depending on the integrity level needs, is considered when providing the uncertainty estimates, which makes them justifiable and more dependable. Compared to naïve approaches, the wrapper also makes uncertainty estimates situation-aware, thereby providing separation between cases with high and low uncertainty (cf. Fig 3). Finally, the framework is model-agnostic as well as holistic in terms of model fit, quality, and uncertainty related to scope compliance.

Based on the Brier score and its components, which we applied to investigate the quality of uncertainty estimates, we showed that an uncertainty wrapper can contribute to estimation quality by improving all the components of the Brier score, i.e., variance, resolution, and unreliability. Specifically, *clustering* cases with a quality impact model as part of an uncertainty wrapper addresses its resolution as well as its *calibration* unreliability. Moreover, a *scope compliance* model, which is also part of the uncertainty wrapper, can help to reduce variance and unreliability by detecting and excluding cases outside the target application scope. In contrast, using a *confidence limit* increases unreliability. However, it also makes overconfident estimates much more unlikely and thus contributes to the overall dependability of uncertainty estimates (cf. Fig. 4).

Although the presented framework was applied and validated in an industry setting with field data, we cannot provide these results in this paper. Instead, we used a publicly available DDM and data generated using a driving simulator. In the next step, however, we plan to provide a more detailed evaluation of the approach on publicly available traffic sign images augmented with typical quality deficits [21] and compare the results with the results of probabilistically extended DDMs.

Acknowledgments. Parts of this work have been funded by the Ministry of Science, Education, and Culture of the German State of Rhineland-Palatinate in the context of the project MInD and the Federal Ministry of Labour and Social Affairs (Feasibility of testing and auditing of Kl-based systems). We would like to thank especially Naveed Akram and Pascal Gerber for providing the dataset we used to illustrate the framework application, and Jan Reich and Sonnhild Namingha for the initial review of the paper.

References

1. M. Kläs: Towards Identifying and Managing Sources of Uncertainty in AI and Machine Learning Models - An Overview. arXiv:1811.11669 (2018).
2. M. Kläs and L. Sembach: Uncertainty wrappers for data-driven models – Increase the transparency of AI/ML-based models through enrichment with dependable situation-aware uncertainty estimates. In: *WAISE 2019*, Turku, Finland (2019).
3. M. Kläs and A. M. Vollmer: Uncertainty in Machine Learning Applications – A Practice-Driven Classification of Uncertainty. In: *WAISE 2018*, Västerås, Sweden (2018).
4. B. Phan, S. Khan, R. Salay and K. Czarnecki: Bayesian Uncertainty Quantification with Synthetic Data. In: *WAISE 2019*, Turku, Finland (2019).
5. Y. Gal: Uncertainty in deep learning. University of Cambridge (2016).
6. M. Henne, A. Schwaiger, K. Roscher and G. Weiss: Benchmarking Uncertainty Estimation Methods for Deep Learning With Safety-Related Metrics. In: *SafeAI 2020*, NY, USA (2020).
7. J. Snoek, et al.: Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems* (2019).
8. A. Niculescu-Mizil and R. Caruana: Predicting Good Probabilities with Supervised Learning. In: *22nd Int. Conf. on Machine Learning* (2005).
9. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel and et al.: Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, vol. 12, 2825-2830 (2011).
10. K. Czarnecki and R. Salay: Towards a Framework to Manage Perceptual Uncertainty for Safe Automated Driving. In: *WAISE 2018*, Västerås, Sweden (2018).
11. Y. Matsuno, F. Ishikawa and S. Tokumoto: Tackling Uncertainty in Safety Assurance for Machine Learning: Continuous Argument Engineering with Attributed Tests. In: *WAISE 2019*, Turku, Finland (2019).
12. C.-H. Cheng, C.-H. Huang and G. Nührenberg: nn-dependability-kit: engineering neural networks for safety-critical systems, https://arxiv.org/abs/1811.06746 (2018).
13. G. W. Brier: Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, 78(1), 1-3, (1950).
14. A. H. Murphy: A new vector partition of the probability score. *Journal of Applied Meteorology*, 12(4), 595-600 (1973).
15. K. Dumicic: Representative Samples. In: Lovric M. (eds) *International Encyclopedia of Statistical Science*. Springer, Berlin, Heidelberg (2011).
16. Developer Survey Results, https://insights.stackoverflow.com/survey/2019 (2019).
17. J. Redmond and A. Farhadi: YOLOv3: An Incremental Improvement, arXiv:1804.02767.
18. A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun: CARLA: An Open Urban Driving Simulator. In: *1st Annual Conference on Robot Learning* (2017).
19. M. Pimentel, D. Clifton, L. Clifton and L. Tarassenko: A review of novelty detection. *Signal Processing*, vol. 99, 215-249 (2014).
20. A. Kumar, P. S. Liang and T. Ma: Verified Uncertainty Calibration. In: *NIPS 2019* (2019).
21. L. Jöckel and M. Kläs: Increasing trust in data-driven model validation. In: *SafeComp 2019*, Turku, Finland (2019).