Probabilistic Constraints Prediction with Uncertainty Quantification in Trajectory Based Operations (TBO)

Paolino De Falco
EUROCONTROL
Bretigny-Sur-Orge, France
paolino.de-falco@eurocontrol.int

Mehtap Karaarslan
EUROCONTROL
Brussels, Belgium
mehtap.karaarslan@eurocontrol.int

***Abstract*—The SESAR research for 4D trajectory data exchanges between Air Traffic Management (ATM) actors enables Trajectory Based Operations (TBO) by introducing more accurate and comprehensive flight data sharing. However, differences in how various ATM actors consider Air Traffic Control (ATC) Letter of Agreement (LoA) constraints, before flight departure, leads to misalignment in trajectory calculations, impacting TBO implementation. To address this problem, as part of the SESAR 3 Network TBO project Solution 1, this paper presents a machine learning-based model predicting the probability for an ATC LoA constraint to be applied during flight. Additionally, an approach to quantifying the predictions' uncertainty is developed, aiming at helping users identify potential issues with predictions. The model outcome will be subject to further assessment via operational validation scenarios in a prototyping environment since it should allow better trajectory alignment across ATM actors, facilitating smoother Flight and Flow - Information for a Collaborative Environment (FF-ICE) Release 1 deployment and advancing TBO in Europe.**

***Keywords*—Machine learning; probabilistic predictions; uncertainty quantification; trajectory constraints; TBO; Trajectory Based Operations; SESAR; FF-ICE**

I. Introduction

One of the key European developments towards TBO was the 4D trajectory data exchanges between all Air Traffic Management (ATM) actors during the pre-departure phase. This development supported the ICAO FF-ICE Release 1 (FF-ICE/R1) definition. ICAO defined the FF-ICE as "*The FF-ICE is flight information-sharing between members of the ATM community. It constitutes the necessary basis for the most advanced ATM systems and the development of four-dimensional (4D) trajectory management*" [1]. The ICAO FF-ICE/R1 eFPL exchange, amongst many other improvements, revolutionises the flight data exchanges worldwide by enabling 4D trajectory (latitude, longitude, altitude, time) data exchange rather than a 2D route (point, time) data exchange, which is in place today.

Exchanging 4D trajectory data enables the different actors to understand how others calculate the flight trajectories. In the past, the SESAR 1, SESAR 2 Wave 1 and Wave 2 solutions which developed the 4D trajectory exchanges between ATM actors and used the data derived from these trajectories identified that there is a misalignment between trajectories calculated by Airspace User (AU), and Network Manager (NM) and Air Navigation Service Providers (ANSPs). This misalignment is negatively impacting the FF-ICE/R1 deployment and achieving TBO in Europe.

One of the reasons of these differences is the different approach in applying the ATC LoA (Air Traffic Control Letter of Agreements) constraints during pre-departure phase which also impacts the post-departure phase. The ATC LoA constraints describe how a flight shall be transferred from one ATC to another by the Air Traffic Controller (ATCO), during the flight execution. The ATC LoA constraint impacts the flight trajectory in the vertical profile, i.e., the flight levels and distances when crossing from one ATC to another.

During the pre-departure phase, the AU does not have to comply with the ATC LoA constraints to obtain a valid flight plan. Hence, the AUs do not systematically consider these constraints in their trajectory calculations. Consequently, since not all ATM actors apply them in the same manner, the resulting different trajectories can cause misunderstandings. Figure 1 comparing the trajectories illustrates this difference. The desired trajectory represents AU 4D trajectory submitted to NM via the FF-ICE Service. The agreed trajectory is the trajectory calculated by NM and distributed to the concerned ANSPs. Since, during the flight the ATCO may or may not decide to apply an ATC LoA constraint entirely, the AUs do not consider them predictable enough to apply systematically during flight planning. The ATC LoA constraints consideration in the AU trajectory calculation increases the fuel that the aircraft shall carry. Therefore, the AU would consider an ATC LoA constraint that the ATCO is most likely to apply.

During the post-departure phase, the trajectory predicted by the aircraft during the flight (Extended Projected Profile - EPP), based on the AU trajectory planned, potentially calculates a different Top Of Descend (TOD) than the one the ATCO expects. This difference prevents certain TBO use cases such as "Descend When Ready via Datalink".

There is a need to solve these problems to complete FF-ICE/R1 deployment throughout Europe; and get closer to the TBO implementation. Since the past SESAR projects identi-

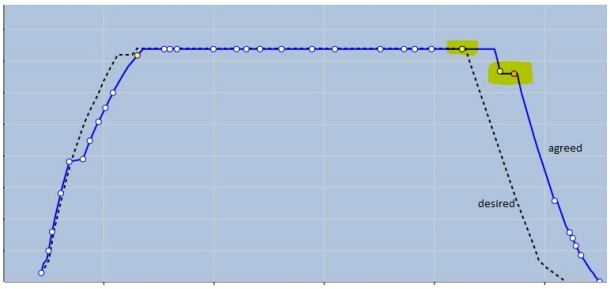


Figure 1. Difference in Top Of Descent (TOD) between desired and agreed trajectories.

fied that **"the AU would consider an ATC LoA constraint that the ATCO is most likely to apply"**, the SESAR 3 Network TBO Project Solution #1 developed a prediction model that is presented in this paper. The purpose of this work is to calculate the probability of application of an ATC LoA constraint along a trajectory by using machine learning (ML) techniques. As a result, the AUs, NM, and ANSPs can predict trajectories better and be aligned in terms of 4D trajectory representation.

The purpose of this paper is to describe the methodology developed (Section III) and its results (Section IV) based on a probabilistic approach with uncertainty quantification (Section II). The paper also discusses the open questions, and the next steps that the project will take, as well as the conclusions (Section V). The "ATC LoA constraint" will be referred to as "constraint" from now on.

II. Uncertainty Quantification Literature Review

ML model metrics offer a broad assessment of a model's overall performance but do not provide insights into the quality or reliability of individual predictions. This implies that, once deployed in an operational environment, the model user can rely on these metrics to assess the overall quality of predictions without having the possibility of determining the reliability of single predictions.

Model predictions and physical observations can be affected by different types of uncertainty. As defined in [2], a common approach is to distinguish between *data uncertainty* (also called aleatory uncertainty), which arises from inherent class overlap or noise within the data, and *knowledge uncertainty* (or epistemic), which stems from a lack of information about input regions that are either far from the training data or sparsely represented in it [3]. This latter form of uncertainty occurs when sampling from a different distribution than the one that generated the training data. High *knowledge uncertainty* may suggest that the model's output should be approached with caution or not trusted. Furthermore, as *knowledge uncertainty* is a property of the model, it can be reduced by providing more training data to the model. *Data uncertainty*, instead, is a property of the underlying distribution that generated the data, rather than a property of the model. A condition for a classification model to capture *data uncertainty* in its predictions is that it is trained using a negative log-likelihood loss function [4]. To be effective, an infinite (or sufficiently large) amount of training data is necessary. It can be shown that prediction with a skewed probability distribution may indicate low *data uncertainty*, whereas a uniform distribution suggests high *data uncertainty*. Models that generalize well tend to produce more accurate estimates of *data uncertainty*, whereas models over-fitting the training data are likely to provide less reliable estimates of this type of uncertainty [4]. *Total uncertainty*, finally, is the sum of both *data uncertainty* and *knowledge uncertainty*.

Various approaches have been proposed in the literature to estimate the uncertainty of individual predictions. Sensitivity analysis methods, for instance, use the model's output response to small perturbations in the input data as a measure of uncertainty [5]. Other methods, such as the Delta method and Bayesian approaches [6], are typically computationally intensive and have been applied for uncertainty quantification. Quantile regression methods [7] enable the simultaneous computation of multiple quantile values, producing a distribution of the target variable for a given input. In this approach, the difference between high and low quantile values can be used to quantify the level of uncertainty for each prediction.

Recently, there has been a growing interest in ATM on quantifying the uncertainty of individual predictions. Several studies have focused on probabilistic predictions of flight delays [8] [9], reactionary delays [8], and the associated costs [10], as well as Air Traffic Flow and Capacity Management (ATFCM) delays [11]. A recent research has focused on probabilistic predictions of aircraft turnaround times [12], while Bayesian probabilistic methods have been applied to estimate trajectory predictions [13] and to guide active learning for meta-modeling in large-scale simulations [14].

Although these studies offer various methods for quantifying uncertainty in individual predictions, the distinction between the two types of uncertainty was not addressed. *Knowledge uncertainty* can be computed by using a single model that learns a decision boundary separating the in-domain region from the rest of the input space, provided that suitable out-of-domain training data is chosen [15], [16]. However, although many of the methods based on single models demonstrate good empirical results, few have a strong theoretical foundation explaining why they are effective [4].

A more intuitive and rigorous approach to compute *knowledge uncertainty* is to use an ensemble of ML models (i.e., a set of models that are trained and combined to solve the same problem) [3]. Ensemble approaches leverage the fact that each single model, behaves predictably within the in-domain data (i.e., input data falling within the feature space covered by the training data) but exhibits undefined behavior for out-of-domain data (i.e., inputs that a ML model encounters during inference that are different from the data it was trained on). As a result, an ensemble of independent models will produce consistent predictions for in-domain inputs and a diverse set of predictions for out-of-domain inputs, since each model's behavior will differ [4]. This property allows an ensemble to assess both *data uncertainty* and *knowledge uncertainty* within

a unified probabilistic framework, eliminating the need for out-of-domain training data which is instead necessary when using single models. In an ensemble of models each model provides a different estimate of *data uncertainty*, which is represented by the entropy of its predictive distribution [4]. The entropy of the averaged probability distribution across all models in the ensemble can serve as a measure of overall prediction uncertainty (or *total uncertainty*). Additionally, by analyzing mutual information, it is possible to decompose the sources of uncertainty and compute *knowledge uncertainty* as the difference between *total* and *data uncertainties*. Equations to compute these quantities for binary classification problems can be found in Section III-D.

In a recent work [3], an approach to quantify and distinguish between the two forms of uncertainties using ensembles of Gradient Boosting based on decision trees (GBDT) models [17] has been developed. Specifically, uncertainty estimation for GBDT models has been analysed by generating an ensemble of models via bootstrapping and using CatBoost as an algorithm [18]. The authors showed that measures of *knowledge uncertainty*, achieved far better out of domain detection performance (assessed using AUC-ROC curves [19]) than measures of *total uncertainty*.

In applications like out-of-domain detection it is desirable to estimate *knowledge uncertainty* (rather than *total uncertainty* [3]). When deployed in an operational environment, ideally, a model could provide not only a probabilistic prediction but also an estimate of the *knowledge uncertainty* associated with it, signaling potential input outliers that could result in undesired predictions or indicating potential issues with the predictions. This additional information can influence the user's decision-making and assist in assigning a certain level of trust to each prediction.

III. Model Development

This section describes a ML binary classification model for predicting the probability that each constraint in a planned trajectory will be applied. Additionally, an approach to quantifying prediction uncertainties will be presented using Catboost [18], one of the Gradient Boosting based on Decision Trees (GBDT) algorithms becoming highly popular in the context of ML for delivering state-of-the-art results on tasks involving heterogeneous features, complex dependencies, and noisy data.

A. Data Preparation & Labeling Approach

The ML model needed to be trained with the historical data including the constraints that were considered on the planned trajectories, and the constraints that were actually applied on the flown trajectories. The NM archives contain two years of planned (with all constraints applied) and flown (updated with radar data) trajectories of the flights occurred in ECAC area. However, this data does not refer to the constraints. Therefore, these constraints had to be identified. In order to identify the constraints on the these trajectories: the planned and flown trajectories from 2023 were retrieved from the NM archives, then the constraints on each trajectory were added by using a tool developed for this purpose. The tool was reprocessing the trajectories by using the NM algorithm to identify the constraints along them. At the end of this data preparation process, 365 days of planned and flown trajectories; and the constraints impacting them were available for training the ML model. Specifically, a trajectory point was labeled with a constraint if, in both the planned and flown trajectories, the same constraint identifier appeared.

Only samples describing trajectory points that were marked with a constraint in the planned trajectory were used for this study. At the end of this data labeling process, 45% of observations were labeled with applied constraints, while 55% with non-applied ones. The final dataset contains nearly ten million observations from 2023 data.

In this study, static input features were used for model development, which enables the application of the model across any time horizon without the need for time-dependent data, enhancing its flexibility across different temporal contexts and prediction intervals. The set of input features that was selected for this study is described in Table I. The point type describes the nature of a trajectory point such as Top Of Descend (TOD). Please access this link for further information.

TABLE I. Attributes used to train the model.

Flight attribute	**Description**	**Convention**
Day of week	Categorical	dayWeek
Month of operations	Categorical	month
Aircraft type	Categorical	aircraftType
Aircraft identifier	Categorical	aircraftId
Aircraft operator	Categorical	aircraftOper
Origin airport	Categorical	origin
Destination airport	Categorical	destination
Trajectory point attribute	**Description**	**Convention**
Latitude of the point	Numerical	latitude
Longitude of the point	Numerical	longitude
Longitude east or west	Categorical	longitudeE-W
Flight level of the point	Numerical	flightLevel
Hour of operations	Categorical	hour
Point type	Categorical	pointType
Constraint identifier	Categorical	constraintId
Point distance	From the origin (nm)	pointDistance

B. Binary Classification with Bootstrapping

CatBoost was chosen for the binary classifier because it efficiently handles categorical data without extensive preprocessing and is robust against overfitting due to its ordered boosting technique [20].

The Log-loss (or negative log-likelihood (NLL) loss) [19] was selected for this model. Log-loss is indicative of how close the prediction probability is to the corresponding actual/true value (0 or 1 in case of binary classification). The more the predicted probability diverges from the actual value, the higher the log-loss value. The choice of Logloss aligns with the goal of obtaining probability estimates, which are essential for evaluating model performance and making informed decisions.

CatBoost allows for tuning multiple hyper-parameters to optimize learning and model performance. In this study, the

focus was on optimizing tree depth and the number of trees, as these significantly impact the loss function. Performance was evaluated using k-fold cross-validation, where the model is trained and tested across different data subsets to calculate an average performance score. To find the best hyper-parameters, GridSearchCV [21] was used, which exhaustively tests all possible combinations to identify the optimal configuration. As a result, the optimal settings were 100 estimators and a maximum tree depth of 7. This analysis was conducted on a single model using the entire dataset, rather than on each individual bootstrapped model.

To enhance the model's robustness, a bootstrap aggregation technique was implemented [22]. Specifically, 50 bootstrap dataset were generated from the original training dataset, with each sample being extracted with replacement. Later, a separate CatBoost model on each bootstrap sample was trained using the previously identified optimal hyper-parameters. A Langevin diffusion-based gradient boosting algorithm [23] was used in CatBoost for training. By adding noise to gradient updates, this method explores the parameter space more broadly, enhancing optimization and reducing over-fitting. It is particularly recommended for the quantification of *knowledge uncertainty* [3]. As a common practice, the dataset was randomly split into a training and testing set using the ratio 80:20, as for all the models presented in this manuscript.

Each of the models within the ensemble outputs two probability values for each of the considered classes. In other words, it provides a probability that a constraint in the planned trajectory will be applied during flight as well as the probability that it will not be applied. Both probabilities are complementary, meaning that their sum is one. Overall, a prediction for a certain observation is considered correct if the model outputs a probability higher than 0.5 for the class that the observation is labeled with. The bootstrapping approach allows for the generation of 50 predictions for each observation that can be used to define new metrics and provide additional information to the potential model user. These metrics will be described in Sections III-C and III-D.

C. Model Performance Assessment Metrics

After training the model on each bootstrap sample, the predicted probabilities of each of the two classes (i.e., applied and non-applied constraints) and of each observation in the test dataset were averaged.

Using these mean probability values and specific threshold values (as it will be shown in Section IV), which allow to classify an instance as belonging to either one class or the other, the overall performance of the binary classification model was assessed according to the accuracy, recall, and precision which are standard metrics for binary classification models [19]. Specifically, an observation is predicted as belonging to a class if the mean value of the predicted probabilities (computed over the ensemble of bootstrap samples) for that class is higher or equal to the threshold value.

Weighted and macro averages of precision and recall will be considered in Section IV-B. Weighted averages adjust metrics based on class proportions, reflecting the influence of each class's size. Macro averages, instead, compute metrics for each class independently and then average them, treating all classes equally.

D. Prediction Uncertainty Quantification Metrics

While accuracy, recall, and precision (described in Section III-C) indicate the overall quality of the model, additional metrics can be used to quantify the uncertainty of each observation within an ensemble, as described in Section II.

Assuming N samples of binary classification models (k = number of classes = 2) obtained via bootstrapping, the *total uncertainty* for each of the j observations can be computed as shown in Equation 1, where log refers to the natural logarithm (the reader might refer to the CatBoost documentation for further details: https://catboost.ai/en/docs/references/uncertainty). Data and *knowledge uncertainty*, instead, can be computed according to the Equations 2 and 3.

$$\text{Total unc.} = -\sum_{k=1}^{2}\left(\frac{1}{N}\sum_{i=1}^{N} Pr_{i,j,k}\right)\cdot log\left(\frac{1}{N}\sum_{i=1}^{N} Pr_{i,j,k}\right) \quad (1)$$

$$\text{Data unc.} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{2}\left(Pr_{i,j,k}\cdot log(Pr_{i,j,k})\right) \quad (2)$$

$$\text{Knowledge unc.} = \text{Total unc.} - \text{Data unc.} \quad (3)$$

Other metrics such as the standard deviation and the difference between the 95th and 5th percentiles of the probabilities from the 50 bootstrapped samples will be also considered as an uncertainty measure since they provide insights into the model's disagreement and variability within an ensemble for each individual observation.

For prediction assessment the Negative Log-Likelihood (NLL) [19] will also be considered. The NLL can be computed as shown in Equation 4, where y_j is the true label of each observation. Lower values of NLL indicate better performance, meaning that the predicted probabilities are closer to the true labels of the binary outcomes while higher NLL values reflect poor model performance.

$$\text{NLL} = -\sum_{k=1}^{2}\left(y_j\cdot log\left(\frac{1}{N}\sum_{i=1}^{N} Pr_{i,j,k}\right)\right) \quad (4)$$

Quantifying the uncertainties and the spread of models' predictions might help to further assess model performance and reliability when deployed in an operational setting. These metrics will be analysed in Section IV-C to provide a comprehensive understanding of the model's performance variability.

IV. Results

This section provides an overview of the binary classifier's performance, starting with an analysis of SHAPley values to evaluate feature importance and contributions. Next, threshold analysis is conducted to optimize decision boundaries and

assess classification performance across various thresholds. Finally, the uncertainty quantification is addressed, focusing on methods to measure and interpret the confidence level of the model's predictions.

A. SHAP analysis

Game theory principles can be applied to interpret the predictions of a ML model by treating each input feature as a player in a game and the model's output as the reward. When input features are included in the model in a random sequence, the contribution of each feature can be determined by calculating the average change in the reward that the existing group of features (or coalition) receives when the new feature joins. This contribution is known as the SHAP (SHAPley) value in the literature [24].

The model initially used for hyper-parameter tuning was utilised for a SHAP analysis. Figure 2 shows SHAP analysis results highlighting the *point type*, *constraint identifier*, and *aircraft identifier* as the most impacting input features.

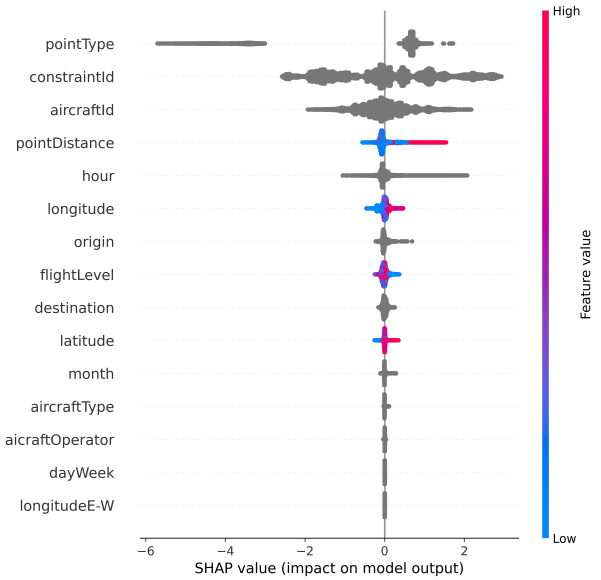


Figure 2. SHAP analysis showing the importance of input features. The vertical axis lists the feature names, arranged from most to least important. Each dot along the horizontal axis represents the SHAPley value of the corresponding feature for a single observation, with the color indicating the feature's magnitude, ranging from blue to red (low to high values). Categorical features are in gray since no magnitude can be associated with them.

B. Threshold Analysis

After training the model, adjusting the threshold value, which represents the probability level at which predictions are classified into one or the other classes, can significantly impact the classifier's performance. As mentioned in Section III-B, the bootstrap method allows to generate 50 probabilistic predictions for each observation. For this analysis, the mean value of the probabilities from the bootstrap samples has been computed to assess the contribution of each observation in the test dataset to the metrics. Figure 3 illustrates the performance metrics of the binary classifier across various threshold values. For this binary classifier, a threshold value of 0.5 appears to balance these metrics optimally as all the metrics reach their maximum. It is expected, indeed, that when a dataset is balanced (i.e., the classes are roughly equal in size), the optimal threshold value for a binary classifier is around 0.5.

The overall model performance, computed with a threshold value of 0.5, is described in Table II. The binary classifier exhibits an accuracy of 0.82, indicating that 82% of the predictions are correct. Both the weighted and macro averages of recall and precision are also ≈ 0.82, suggesting that the classifier performs uniformly across different classes.

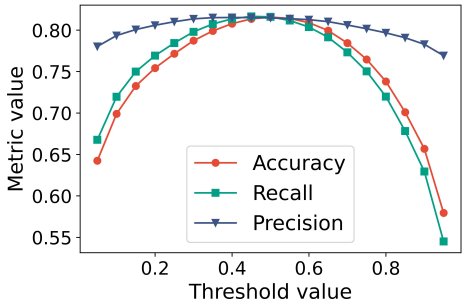


Figure 3. Accuracy, recall and precision as a function of the threshold values

TABLE II. MODEL PERFORMANCE METRICS COMPUTED ON THE TEST SET

Macro average (weighted average)		
Precision	**Recall**	**Accuracy**
0.81 (0.82)	0.82 (0.82)	0.82

C. Knowledge, Data and Total Uncertainty

Until now, the overall model performance was described. This section focuses on the quantification of the three types of uncertainty: *total*, *data* and *knowledge* uncertainties. Specifically, for each observation, these quantities have been computed according, respectively, to the Equations 1, 2, 3.

Figure 4 illustrates the model's performance over 10000 samples randomly selected from the test dataset, evaluated using the NLL and three types of uncertainties. For this analysis, a kernel density estimation (KDE) on the data was performed [25]. Specifically, the density of data points was computed using a Gaussian function as a kernel and the Scott's rule [26] for the bandwidth estimation. In Figure 4, it is possible to observe that not only the overall trends but also the orders of magnitude of total and data uncertainties are comparable while the quantities describing the *knowledge uncertainties* are relatively small. This outcome is not surprising since *knowledge uncertainty* was computed on in-domain observations sampled from the testing dataset which belongs to the same distribution of the training dataset. It is, instead, expected that the *knowledge uncertainty* will increase when tested on out-of-domain observations. When the NLL is relatively low (indicating correct predictions) both the *total* and *data uncertainty* can vary significantly, showing that the

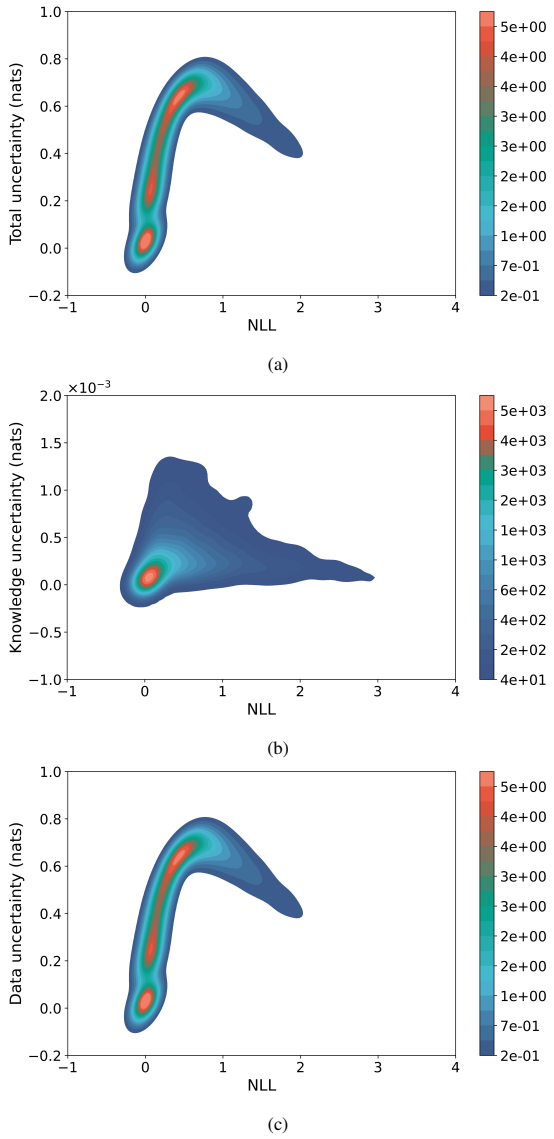


Figure 4. The relationship between NLL and different types of uncertainties, illustrated using kernel density estimation (KDE). (a) Total uncertainty vs NLL, showing the overall uncertainty in model predictions. (b) Knowledge uncertainty vs NLL, highlighting uncertainty due to lack of model knowledge. (c) Data uncertainty vs NLL, reflecting inherent uncertainty in the data.

noise in the data is uniformly spread. *Knowledge uncertainty* remains close to zero for correct predictions and does not increase significantly for incorrect predictions, at higher NLL values.

Analyzing the results while differentiating between correct and incorrect predictions could offer additional insights. Figures 5a and 5b show that, in case of correct predictions, both the *total uncertainty* and the *knowledge uncertainty* tend to increase when the model output for the class that was correctly predicted is close to 0.5, while they are relatively low for outputs close to one. For the sake of simplicity, plots for the *data uncertainty* have been omitted since it has been previously observed (in Figure 4) that both *data uncertainty* and *total uncertainty* exhibit similar behavior. For incorrect predictions, both types of uncertainty tend to increase when the model output is close to 0.5 (Figures 5c and 5d).

Additionally, understanding how the percentile differences (computed according to the description provided in Section III-D) relate to the uncertainty values could provide further indication on model performance. Figure 6 reveals a linear relationship between *knowledge uncertainty* and the percentile difference for both correct and incorrect predictions. In contrast, *total uncertainty* does not exhibit a clear pattern, making it difficult to draw similar conclusions.

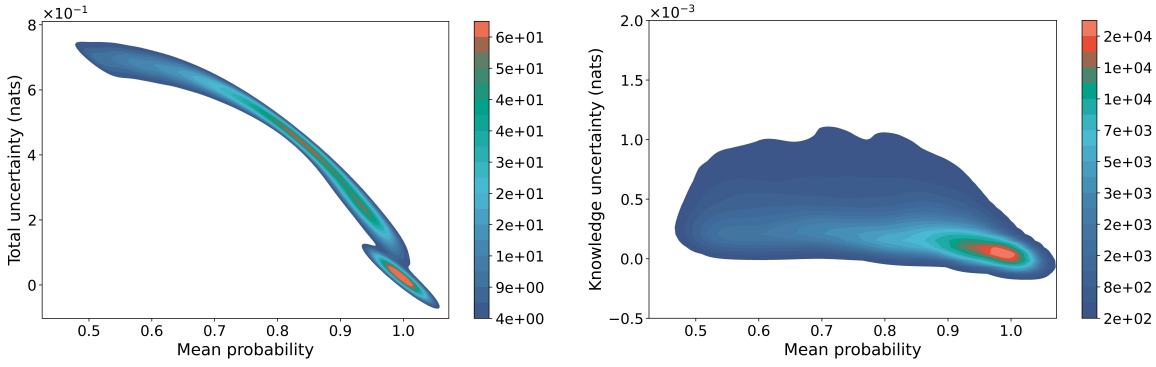
Interestingly, for correct predictions, the percentile differences tend to reduce when the model outputs probabilities near one (when displaying the values of the correctly predicted class), while for incorrect predictions, the percentile differences tend to increase when the model outputs probabilities close to 0.5 (when displaying the values of the incorrectly predicted class). Similar conclusions could be extended to the standard deviation of the predictions within the ensemble, as it was found that it linearly correlates with the the percentile differences, in case of both correct and incorrect predictions.

V. Discussion & Conclusions

ML model labeling is crucial, as it can influence the model's performance and results. In this work, the labeling process relies on the constraints identification in planned and flown trajectories that contain the same identifier. This approach is expected to be effective when there are small differences (in terms of time and distance) between the two trajectories, i.e. the trajectories are comparable. In reality, there might be substantial differences, e.g. due to rerouting or longer delays. Such planned and flown trajectories become incomparable. In these cases, the labeling is challenging and requires further analyses. This algorithm needs to evolve to identify comparable versus incomparable trajectories.

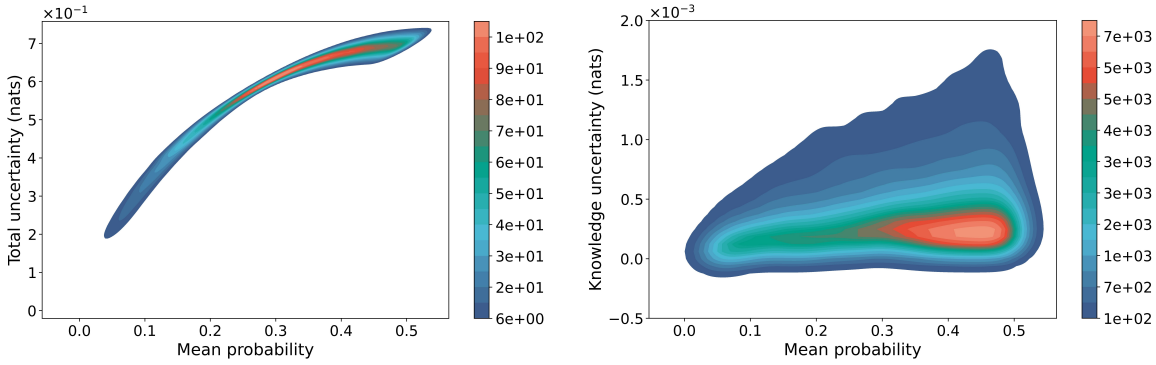
ML models need a set of input data during inference. A crucial factor is ensuring that this data is available. For this work's model, only static features have been used as inputs, ensuring that the model remains usable at any time. In the future, for e.g., one might include the weather forecast indicators which might impact the constraints application and, therefore, model's performance.

In binary classification, selecting the threshold value is important as it determines the probability level for classifying predictions. For the presented binary classifier, a threshold of 0.5 optimally balances performance metrics, which is expected for balanced datasets. Another aspect to consider regarding the presented approach is the hyper-parameter search, which was conducted using GridSearchCV over a predefined set of parameters. This analysis was performed on a unique model using all the available data and not for each individual bootstrapped model, based on the assumption that the variations in model inputs across bootstrapped samples do not necessitate different hyper-parameter settings. Future work could explore the impact of tuning hyper-parameters individually for each model within the ensemble, although this approach may be computationally intensive.



(a) Correct predictions

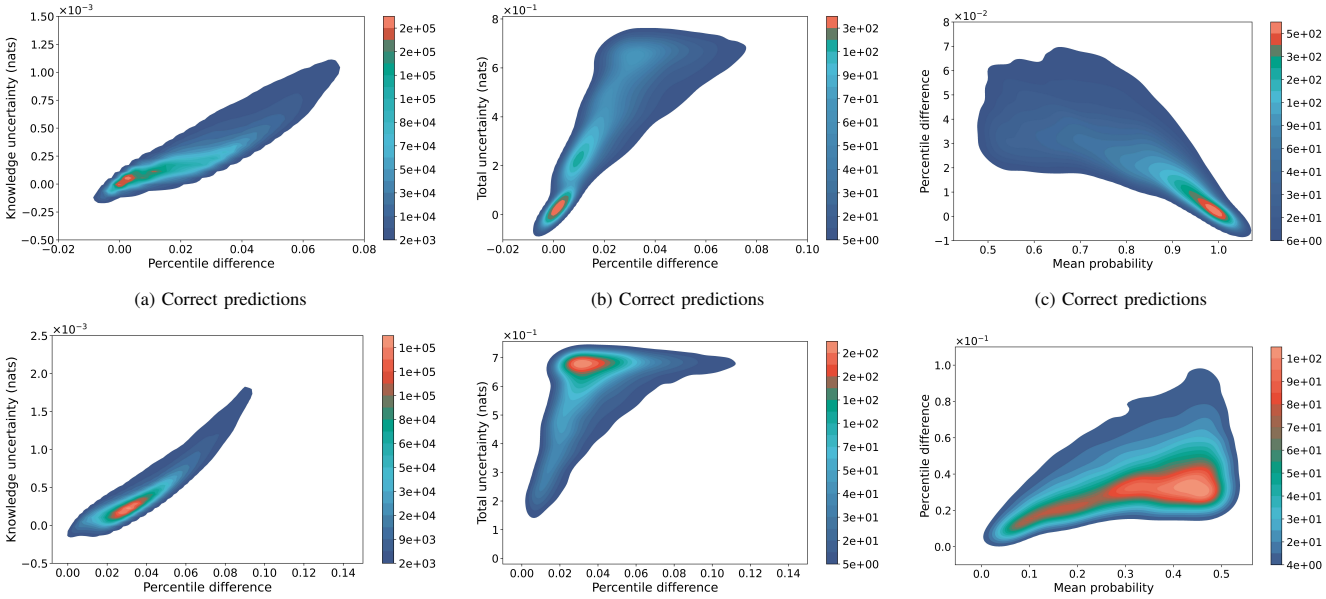
(b) Correct predictions



(c) Incorrect predictions

(d) Incorrect predictions

Figure 5. Total and knowledge uncertainty as a function of the mean probability values computed for each observation using the ensemble of 50 models. Figures (a) and (b) show the results for correct predictions, while figures (c) and (d) display the results for incorrect predictions. Only the values for the correctly predicted and for the incorrectly predicted classes have been considered, respectively, in figures (a)-(b) and (c)-(d).



(a) Correct predictions

(b) Correct predictions

(c) Correct predictions

(d) Incorrect predictions

(e) Incorrect predictions

(f) Incorrect predictions

Figure 6. Total and knowledge uncertainty as a function of the difference between the 5th and 95th percentiles of the probability values computed for each observation using the ensemble of 50 models. Figures (a), (b) and (c) show the results for correct predictions, while figures (d), (e) and (f) display the results for incorrect predictions. Only the values for the correctly predicted and for the incorrectly predicted classes have been considered, respectively, in figures (a)-(b)-(c) and (d)-(e)-(f).

Quantifying model uncertainty can improve decision-making by providing a clearer understanding of prediction confidence. Additionally, it can support better risk management by estimating potential errors and guiding actions based on the level of uncertainty in predictions. In this work, an approach to quantify *total*, *data* and *knowledge uncertainties* has been proposed. It has been shown that the three types of uncertainty follow similar patterns: they are generally high when the model outputs probabilities around 0.5 and lower when probabilities are near 1 or 0, depending on whether the predictions are, respectively, correct or incorrect (please note that in Figures 4 and 5 either the values for the correctly predicted or for the incorrectly predicted classes have been shown). Interestingly, the percentile range, computed as the difference between the 95th and 5th percentiles of the probability distributions, exhibits a similar trend to knowledge uncertainties (Figure 6). This suggests that the percentile range could potentially replace the knowledge uncertainty metric. A similar observation applies to the standard deviation of probability values in the ensemble, as the standard deviation and the percentile difference exhibit a linear relationship (Section IV-C).

It is important to note that the magnitude of *knowledge uncertainties* values is significantly lower compared to the other two types of uncertainty, indicating that *data* uncertainty is the predominant factor. This result was expected since the *knowledge uncertainties* has been computed based on the observations within the test dataset, which is part of the in-domain-distribution of data. In the future research, out-of-domain data could be generated to more accurately quantify the trends and magnitude of *knowledge uncertainty* that users might encounter when the model is provided with specific types of out-of-domain input data. Various approaches have been developed for this task in the field of generative Artificial Intelligence (AI) [27]. *Knowledge uncertainty* values could be converted into colors or other forms of representation to make their interpretation easier. This will help the user identifying potential input outliers or issues with predictions, influencing the decision-making and the level of trust.

These results are inputs to the discussions with the operational experts about how this information can be used. To facilitate these discussions, a web based tool is developed to visually compare trajectories and see the constraint probability by using real-time operational data or test data from NM systems (Figure 7).

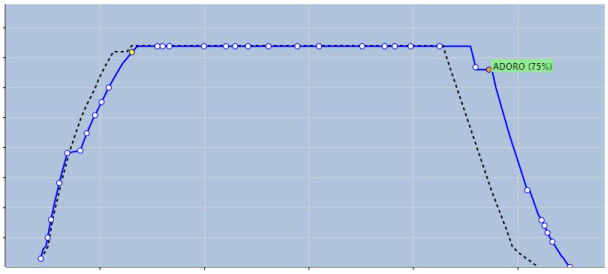


Figure 7. Agreed trajectory with a constraint probability vs flown trajectory.

With the tool and ML model results, the experts will be looking for answers to the following questions:

- Is the probability acceptable for AU use? What would be the threshold probability to apply a constraint?
- If and which other ATM actors can use the probability?
- If and how the uncertainty can be used?

And quantify benefits by answering the following questions:

- Will the AU trajectory be more predictable?
- Will the ATCO and Pilot workload will decrease?
- Will the ATM actors' trajectories be more aligned?

References

[1] "Manual on flight and flow information for a collaborative environment (ff-ice)," *ICAO, Doc 9965 AN/483*, 2012.

[2] Y. Li, J. Chen, and L. Feng, "Dealing with uncertainty: A survey of theories and practices," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, pp. 2463–2482, 11 2013.

[3] A. Malinin, L. Prokhorenkova, and A. Ustimenko, "Uncertainty in gradient boosting via ensembles. arxiv 2020," *arXiv preprint arXiv:2006.10562*.

[4] A. Malinin, "Uncertainty estimation in deep learning with application to spoken language assessment," Ph.D. dissertation, 2019.

[5] Z. Bosnić and I. Kononenko, "Estimation of individual prediction reliability using the local sensitivity analysis," *Applied intelligence*, vol. 29, pp. 187–203, 2008.

[6] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "Comprehensive review of neural network-based prediction intervals and new advances," *IEEE Transactions on neural networks*, vol. 22, no. 9, pp. 1341–1356, 2011.

[7] S. J. Moon, J.-J. Jeon, J. S. H. Lee, and Y. Kim, "Learning multiple quantiles with neural networks," *Journal of Computational and Graphical Statistics*, vol. 30, no. 4, pp. 1238–1248, 2021.

[8] R. Dalmau, P. De Falco, M. Spak, and J. D. R. Varela, "Probabilistic pretactical arrival and departure flight delay prediction with quantile regression," *Journal of Air Transportation*, vol. 32, no. 2, pp. 84–96, 2024.

[9] M. Zoutendijk and M. Mitici, "Probabilistic flight delay predictions using machine learning and applications to the flight-to-gate assignment problem," *Aerospace*, vol. 8, no. 6, p. 152, 2021.

[10] P. De Falco and L. Delgado, "Prediction of reactionary delay and cost using machine learning," 2021.

[11] S. Mas-Pujol, L. Delgado, and P. De Falco, "Pre-tactical advice using machine learning for air traffic flow management delay estimation," *Airline Group of the International Federation of Operational Research Society (AGIFORs)*, 2022.

[12] P. De Falco, J. Kubat, V. Kuran, J. Rodriguez Varela, S. Plutino, and A. Leonardi, "Probabilistic prediction of aircraft turnaround time and target off-block time," *13th SESAR innovation days. Seville, Spain*, 2023.

[13] R. Graas, J. Sun, and J. Hoekstra, "Quantifying accuracy and uncertainty in data-driven flight trajectory predictions with gaussian process regression," *11th SESAR Innovation Days*, 2021.

[14] C. Riis, F. Antunes, G. Gurtner, F. C. Pereira, L. Delgado, and C. M. L. Azevedo, "Active learning metamodels for atm simulation modeling," in *11th SESAR Innovation Days*, 2021.

[15] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," *arXiv preprint arXiv:1711.09325*, 2017.

[16] A. Malinin, A. Ragni, K. Knill, and M. Gales, "Incorporating uncertainty into deep learning for spoken language assessment," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2017, pp. 45–50.

[17] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[18] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[20] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: gradient boosting with categorical features support," *preprint arXiv:1810.11363*, 2018.
[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
[22] C. Z. Mooney, R. D. Duval, and R. Duvall, *Bootstrapping: A nonparametric approach to statistical inference*. sage, 1993, no. 95.
[23] A. Ustimenko and L. Prokhorenkova, "Sglb: Stochastic gradient langevin boosting," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 487–10 496.
[24] L. S. Shapley *et al.*, "A value for n-person games," 1953.
[25] S. Weglarczyk, "Kernel density estimation and its application," in *ITM web of conferences*, vol. 23. EDP Sciences, 2018, p. 00037.
[26] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.
[27] P. Marek, V. I. Naik, V. Auvray, and A. Goyal, "Oodgan: Generative adversarial network for out-of-domain data generation," *arXiv preprint arXiv:2104.02484*, 2021.