Prediction of Flight Departure and Arrival Routes with Gradient Boosted Decision Trees

A Case Study for Amsterdam-Schiphol Airport

Amine Heffar, Ramon Dalmau & Eric Allard
Network (NET) Research Unit
EUROCONTROL Innovation Hub (EIH)
Brétigny-Sur-Orge, France

***Abstract*—Accurate trajectory predictions are of paramount importance to obtain representative traffic demand figures, and therefore to apply effective and efficient air traffic flow and capacity management measures. The trajectory of an aircraft can be divided in (1) the departure phase from the origin, (2) the en-route phase, and (3) the arrival phase to the destination. For the departure and arrival phases, air traffic controllers must provide the clearance to execute one of the routes pre-defined by the local authorities. The route that is assigned to an aircraft depends on several factors, such as the city-pair, the runway configuration, and environmental restrictions (e.g., those applied to mitigate noise in the surrounding communities). This paper proposes a machine learning model that, trained with historical data, is able to capture the influence of these factors and accurately predict which departure and arrival routes will be executed by a particular flight well before take-off and landing, respectively. Promising results of a case study for Amsterdam-Schipol airport using traffic and weather data from 2019 show that the model is able predict departure routes with an accuracy of 0.93 four hours before take-off, and arrival routes with an accuracy of 0.87 five hours before landing. Furthermore, a comprehensive feature importance analysis reveals which are the most important factors that determine the departure and arrival routes of an aircraft, therefore allowing to interpret the predictions of the model.**

***Keywords*—Machine learning; trajectory prediction**

I. Introduction

In many airports with instrumental flight rules procedures (IFR), departing and arriving flights follow standard routes designed by local authorities. The goal of these routes is to enable safe and efficient flow of traffic in the terminal manoeuvring area (TMA), as well as to standardise and simplify the description of departure and arrival procedures in air traffic control (ATC) clearances, which would otherwise require long radio transmissions between the pilot and ATC.

A standard instrument departure (SID) consists of a sequence of flight *legs*, which describes in detail how aircraft should proceed from the runway threshold to a particular waypoint en-route. Analogously, a standard arrival route (STAR) describes how aircraft should proceed from a particular waypoint en-route to the first waypoint of the approach to the airport, e.g., the initial approach fix. For each airport, several SIDs and STARs are typically designed, aiming to effectively accommodate a large variety of traffic patterns, aircraft types, navigation capabilities, weather conditions, and/or environmental restrictions.

In most of Europe, SIDs are named after the final waypoint of the procedure, followed by a version number that is incremented by one each time the procedure is modified, and a single letter that designates the associated runway. For instance, LOPIK1F refers to a SID of Amsterdam-Schipol airport (EHAM) that connects runway 04 with the LOPIK waypoint. For STARs, the equivalent nomenclature is used, but the initial waypoint of the procedure is used instead. In United States, however, a single SID or STAR may serve multiple runways. Accordingly, the last letter is omitted from its name.

During the preparation of the flight plan, airspace users can specify the preferred SID and STAR. The actual SID, however, is assigned by the ATC during the departure clearance, right before take-off. Similarly, the actual STAR is communicated to the pilot shortly before the top of descent. Accordingly, the SID and STAR that are planned by the airspace user may differ from those executed, depending on the weather conditions, the runway configuration in use at the origin (resp. destination) airport, or the need to separate air traffic, among other factors.

A last-minute change of SID and/or STAR may have a significant impact on the overall trajectory, and so on the sequence of airspace sectors (and corresponding entry times) crossed by the flight. From the network point of view, accurate predictions of the SIDs and STARs several hours in advance would reduce the uncertainty in the air traffic demand, and allow for more informed, efficient and effective demand and capacity balance measures. From the airport point of view, accurate predictions of SIDs and STARs could aid the departure sequencing tools for collaborative decision making and therefore increase the runway throughput. From the environmental perspective, accurate predictions of SIDs and STARs could be used to inform the local communities around the airports about the expected noise levels in the next hours. The fundamental objective of this work is to enable these benefits.

In the recent years, a wide variety of techniques have been proposed to identify the main departure and arrival flows in the TMA from surveillance data (e.g., automatic dependent surveillance-broadcast), including hierarchical clustering [1], spectral clustering [2], classical machine learning techniques [3]–[5], as well as deep learning [6]. Other works used similar techniques and data, but aiming to determine the most characteristic routes between city-pairs [7], [8].

Needless to say that the identification of major departure and arrival routes (or flows) has received a lot of attention in the literature, yet predicting which of these routes a given aircraft will execute has not been as popular. In this context, [9] proposed an artificial neural network to predict the actual trajectory that will be flown by an aircraft in the Maastricht upper area control centre (MUAC) area of responsibility, [7] addressed the route prediction problem between city-pairs by using decision trees, [10] solved the same problem by combining recurrent and convolutional neural networks, and [11] proposed a model that creates multiple possible routes in the TMA and infers their probability distribution.

This paper tackles the problem of predicting (several hours in advance) the SID and STAR that a flight will execute. Instead of relying on complex hard-coded rules meticulously designed by domain experts, state-of-the-art machine learning techniques have been used to train a model on a large amount of historical traffic and weather data. The model proposed in this paper is based on ensemble methods, which have demonstrated outstanding performance in many classification tasks using structured data [12]. The predictors used by the model to determine the actual SID or STAR include: the expected route as reported by the Enhanced Tactical Flow Management System (ETFMS), context features (e.g., airline, city-pair), weather features (e.g., wind direction), features related to the traffic load at the airport, as well as standard calendar features.

Results from a case study with flights departing and arriving at EHAM during 2019 show that the model is able to accurately predict the SID and STAR that will be executed by each flight. The performance of the model is compared with a baseline predictor that uses the expected route of the ETFMS as the most accurate prediction. Last but not least, a comprehensive feature importance analysis on the trained model reveals the most influencing predictors.

II. Background

Once a flight has departed (resp. arrived) the actual SID (resp. STAR) can be obtained from the data distributed by the enhanced tactical flow management system (ETFMS). Before departure (resp. arrival), the expected SID (resp. STAR) can be also obtained from these data. Despite being the best and most up-to-date information available nowadays, the accuracy of the SID and STAR expected by the ETFMS (as any prediction of the future) degrade with the look-ahead time. The large amount of historical data about which SID and STAR were executed under certain conditions (e.g., wind direction, aircraft type, hour of the day) encourage the use of *supervised* machine learning techniques to learn from the past and reduce the uncertainty of the future. Section II-A shows the accuracy of the expected SID and STAR as reported by the ETFMS for flights departing and arriving from/to EHAM during 2019. Section II-B describes the fundamentals of the model trained in this paper to improve such accuracy.

A. The current route predictions

The ETFMS is the supporting tool for coordination between ATC, air traffic flow and capacity management (ATFCM), airports and aircraft operators (AOs), which provides global data to be shared among the different stakeholders. As such, the ETFMS collects data from a variety of sources: initial flight plan processing system (IFPS) messages; aircraft communications, addressing and reporting systems (ACARS); departure planning information (DPI) from airports which have implemented collaborative decision-making (CDM) and advanced ATC tower airports; correlated position reports (CPRs), etc.

The ETFMS flight data (EFD) messages provide the most up-to-date state of each flight from the submission of the initial flight plan (typically 6 to 9 hours before take-off) to flight termination. Each EFD message of a flight includes static information such as the departure and destination airports, the AO, etc., as well as dynamic information like the status of the flight (e.g., airborne), the off-block and arrival times, the air traffic flow management (ATFM) delay, etc. EFD messages are distributed at certain events (e.g., submission of initial flight plan, change of ATFM delay) and also periodically [13].

Each EFD message also includes the expected SID and STAR. These predictions are not static, but evolve as the ETFMS has access to more accurate information about the route of the flight plan, the runway configuration at the departure/arrival airports, the departure/arrival sequences, etc. As the take-off and arrival time approaches, these forecasts converge to the actual SID and STAR, respectively.

Figure 1 shows the accuracyof the SID and STAR reported in the EFD, as a function of the look-ahead time. This figure has been computed by considering all flights that departed and arrived from/to EHAM during 2019.

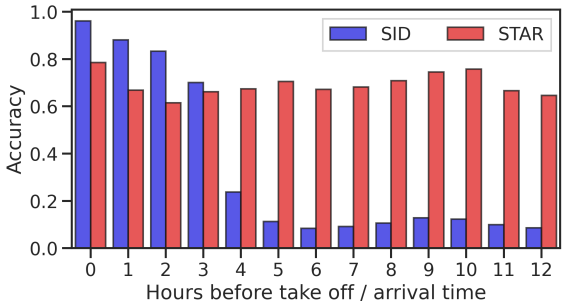


Figure 1: Accuracy of the SID and STAR reported in the EFD for flights departing/arriving from/to EHAM (2019)

According to Fig. 1, 1 hour or less before take-off, the accuracy of the SID reported in the EFD for flights departing from EHAM during 2019 was higher than 90%. This remarkable accuracy gradually deteriorates until 3 hours before take-off, when it is 70%. A sudden drop to 20% is observed 4 hours before take-off. Thereafter, the accuracy remains fairly stable.

This behaviour could be explained by the fact that, far from take-off time, the SID reported by the ETFMS relies on the flight plan information, which does not consider potential runway configuration changes due to, e.g., noise abatement.

Results shown in Fig. 1 suggest that the maximum benefit of a machine learning model would be obtained at, approximately, 4 hours before take-off. Earlier than this time, the predictions of the model would be too uncertain; and later than this time, the relative benefit with respect to the ETFMS and the room for action would be minor. Thus, the machine learning model has been trained to perform SID predictions 4 hours before the expected take-off time (ETOT).

Interestingly, the accuracy of the STAR reported in the EFD shows a very different distribution. According to Fig. 1, the accuracy of the STAR reported in the EFD for flights arriving to EHAM during 2019 ranged from 60% to 80%. Despite the accuracy 1 hour or less before arrival is the highest, a trend with the look-ahead time cannot be observed. In this study, the machine learning model has been trained to perform STAR predictions 5 hours before the estimated time of arrival (ETA).

B. Gradient boosted decision trees

Let us define $\boldsymbol{x} \in \mathcal{X}$ as the (input) features vector and $y \in \mathcal{Y}$ the (output) target, where $\mathcal{X}$ and $\mathcal{Y}$ are the features and target spaces, respectively. Given a finite training set $\mathcal{T} = \{(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}\}$, a supervised learning algorithm aims to find the *best* (presumably nonlinear) model f to predict the unknown $y \in \mathcal{Y}$ for any observed $\boldsymbol{x} \in \mathcal{X}$. Note that the objective of supervised learning is not to perfectly fit f to $\mathcal{T}$ (otherwise a classical look-up table would suffice) but to obtain a model with the ability to *generalise* and perform accurate predictions for samples not seen during the training process.

Several combinations of models and algorithms can be used to solve supervised learning tasks, ranging from simple logistic regression models to complex artificial neural network architectures. The ability of the model to generalise depends on the task to be accomplished, the amount and the quality of the data, the model and its *hyper*-parameters (e.g., the number of neurons or layers in case of neural networks).

In any case, the model f can be described with a set of unknown parameters Θ, which are adjusted during training to minimise the expected value of a loss function L on $\mathcal{T}$:

$$\arg\min_{\Theta} \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} L\left(y, f\left(\boldsymbol{x}, \Theta\right)\right). \tag{1}$$

For regression tasks (i.e., when y is a continuous variable), L could be simply the absolute difference between the predicted and the actual target, i.e., $L(y, f(\boldsymbol{x}, \Theta)) = |y - f(\boldsymbol{x}, \Theta)|$. For classification tasks (i.e., when y is a discrete variable), the cross-entropy loss is typically used:

$$L\left(y, f\left(\boldsymbol{x}, \Theta\right)\right) = -\sum_{i\in\mathcal{Y}} \mathbb{1}_y(i) \log\left(p_i \,|\, f\left(\boldsymbol{x}, \Theta\right)\right), \tag{2}$$

where the indicator function $\mathbb{1}_y\left(i\right)$ is:

$$\mathbb{1}_y\left(i\right) := \begin{cases} 1 & \text{if } i = y \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

and p_i is the (predicted) probability that the sample which characteristics are described by $\boldsymbol{x}$ belongs to the class $i \in \mathcal{Y}$.

The model f proposed in this paper to predict SID and STAR is based on ensemble methods, which are a collection of machine learning techniques that construct a strong learner from a set of weak learners. *Boosting* is a well-known ensemble method, which consists of iteratively training a sequence of weak learners, with the training examples for the next learner weighted according to the accuracy of the previously created learners. Gradient boosting decision trees (GBDT) is a model trained by using the boosting method on a sequence of decision trees (the weak learner). In particular, at each iteration a new decision tree is created by fitting the gradients of the residuals of the previous decision trees.

Because of their cutting-edge performance in a variety of practical applications, GBDT are popular models in the field of machine learning. Traditional GBDT implementations, such as XGBoost [14], are suitable for a wide range of practical applications, but their efficiency and scalability suffer when the number of features and/or samples in the train set are huge. The fundamental reason is that, for each feature, they need to evaluate all the samples in order to estimate the information gain of all possible split points, which might take a long time.

LightGBM [15] is another GBDT implementation that adopts gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB) to overcome the limitations of XGBoost. By removing a considerable fraction of samples with small gradients, the former strategy reduces the number of evaluations needed to compute the gain information at each iteration. The latter method is used to minimise the number of features by grouping together those that only take non-zero values on rare occasions. In this paper, the lightGBM[1] implementation has been selected as the machine learning model for predicting departure and arrival routes.

It is beyond the scope of this work to go into the lightGBM implementation in detail. The reader is referred to the original publication for more information on the peculiarities of lightGBM [15], or a previous work in which this model was used to improve the prediction of take-off times [16].

III. Model

In this paper, a GBDT model (see Section II-B) has been trained with historical data to improve SID and STAR predictions (see Section II-A). Section III-A describes the output space $\mathcal{Y}$ predicted by the model, and Section III-B describes the input space $\mathcal{X}$ used by the model to make predictions.

A. Output target

Predicting the SID (or STAR) that will be cleared to an aircraft can be addressed as a multi-class classification problem, in which the model f has to estimate the most likely procedure from a finite, pre-defined, set $\mathcal{Y}$. In this case, the finite set corresponds to the catalogue of possible SIDs (or STARs) published for the airport subject of study.

[1]Note that, in the following sections, the authors call the machine learning model GBDT and not lightGBM for the sake of simplicity.

As mentioned in Section I, in most of Europe SIDs and STARs are runway-specific. Consequently, there may exist as many SIDs ending at a specific waypoint as runways in the airport, and each one of these SIDs will have a different name. Note that the analogous statement applies for the STARs. At the moment of writing this paper, the number of SIDs published at EHAM, for instance, is 89, which end at 23 waypoints. For example, there exist 7 routes (ANDIK1N, ANDIK1S, ANDIK1T, ANDIK2E, ANDIK2F, ANDIK2G, ANDIK2R) ending at the waypoint ANDIK, and each one is associated with a different runway.

In this study, the runway dependence has not been taken into account. Actually, model has been trained to predict the last waypoint of the SID (resp. first waypoint of the STAR). The main reason of this simplification is that, for the fundamental objective of this work discussed in Section I, taking into account the specific runway associated to the SID or STAR would significantly increase the complexity of the problem (because the model would have a much larger target space $\mathcal{Y}$ to select from) in return for a minor operational benefit.

Remember from Section II-A, that each flight may send several EFD messages from the submission of the initial flight plan to termination, which are triggered by specific events or periodically. In this paper, only the last EFD message of each flight before the selected look-ahead time has been used.

Summing up, the model presented in this paper was trained to predict the last waypoint of the SID 4 hours before ETOT, and the first waypoint of the STAR 5 hours before ETA, using the most up-to-date information available at that time. Last but not least, it should be noted that two GBDT models have been trained independently: one to predict the SID, and one to predict the STAR. As discussed in the next section, however, the set of input features for the two models is equivalent.

B. Input Features

In a nominal situation, the SID (or STAR) that is cleared by the ATC relies mainly on the flight attributes (e.g., the city-pair), the traffic load, the weather conditions and/or environmental restrictions. In turn, in some airports the latter are applied during specific periods of time. The model presented in this paper was designed by taking into account these factors.

Table I lists the input features used by the model to predict the SID (resp. STAR) of a given flight. These features have been classified into four main categories according to their nature. Furthermore, whether the feature is numerical (i.e., continuous) or categorical (i.e., discrete) is also indicated.

All the features belonging to the flight attributes category are extracted from the latest EFD message received before performing the prediction. These features provide basic information about the flight for which the SID (or STAR) is predicted. Note that the destination airport is used by the model that predicts the SID, while the model that predicts the STAR requires the departure airport. Analogously, the expected SID as reported in the EFD message is used by the model that predicts the SID, while the model that predicts the STAR requires the expected STAR.

TABLE I: Input Features

Category	Feature	Type
Flight	Destination (resp. departure) airport Expected SID (resp. STAR) Aircraft type Aircraft operator	Categorical
	Expected taxi time	Numerical
Weather	Sky condition (OVC, SCT, FEW, BKN, ...) Wind direction (N, NNE, NE, ENE, E, ...)	Categorical
	Visibility Temperature Dew point Pressure Wind speed	Numerical
Calendar	Month Day of the week Hour of the day	Categorical
Traffic	Departure traffic demand Arrival traffic demand Traffic demand for each SID (resp. STAR)	Numerical

The weather features have been extracted from meteorological terminal aviation routine weather reports (METARs), which describe the current weather conditions at the airport using a standard format. In most of the major airports, METARs are generated once an hour. In this work, the weather features have been extracted from the latest METAR available before performing the prediction by using the python-metar library[2].

Furthermore, calendar features have been included to help the model in capturing (potential) seasonal patterns.

Finally, the traffic load features have been computed by using the most up-to-date ETFMS data about all the flights that departed from (or arrived to) EHAM recently. In particular, the traffic demand is computed as the number (i.e., counts) of flights that departed or arrived during the last two hours before the moment when the prediction is performed. According to Table I, each model is fed with $|\mathcal{Y}| + 2$ features related to the traffic load: one that captures the total departure traffic demand at the airport, one that captures the total arrival traffic demand at the airport, and one that captures the departure (resp. arrival) traffic demand for each individual SID (resp. STAR) in the catalogue of possible choices. Note that the model that predicts the SID does not consider the arrival traffic demand for each individual STAR for the sake of simplicity. The analogous statement applies for the STAR model.

It is important to mention that the input features listed in Table I are not tailored to EHAM. These features are generic and could be collected for any other airport to train a *copy* of the proposed model and predict the associated procedures.

IV. Results

The model presented in Section III was trained and optimised by using thousands of samples collected from historical traffic and METAR data. During training, the *hyper*-parameters of the model were optimised to boost its predictive power.

[2]https://github.com/python-metar/python-metar

Then, the performance of the *best* model was evaluated on a set of samples not seen during training (i.e., the test set). Finally, a comprehensive feature importance analysis based on additive feature attribution methods (AFAM) was performed to explain its predictions. Essentially, AFAM methods assign an importance score $\phi_i(\boldsymbol{x})$ to each feature $i \in \mathcal{X}$. The sum over $\phi_i(\boldsymbol{x})$ approximates the output $f(\boldsymbol{x}, \Theta)$ of the model for the sample $(\boldsymbol{x}, y)$. In this paper, the Shapley method (SHAP) [17] was used to determine the attribution of each input feature for the prediction of each sample $(\boldsymbol{x}, y)$ in the test set.

Section IV-A describes the the training process and lists the best *hyper*-parameters of the model. An illustrative example is provided in Section IV-B. Finally, Sections IV-C and IV-D show the aggregated performance metrics on the test set.

A. Training and optimisation process

The flight traffic data used in this paper corresponds to the EFD messages sent for all flights that departed and arrived from/to EAHM during 2019; while the weather data was obtained from all the METARs of EHAM during 2019.

The dataset consists of ∼11M EFD messages corresponding to ∼250K departing flights, ∼10M EFD messages corresponding to ∼233K arriving flights, and ∼18K METARs. Note that the SID model was trained with the departing flights, while the arriving flights were used to train the STAR model.

Remember from Section III-A that, for each one of the flights, only the last message before the selected look-ahead time (4 hours before ETOT for the SID model, and 5 hours before ETA for the STAR model) was used. Consequently, the number of EFD messages is the same as the number of flights.

For both SID and STAR models, the typical 80%-20% train-test split was used. In other words, 80% of the samples were randomly selected for training the model and fine-tuning its *hyper*-parameters, and the remaining 20% were reserved to assess the performance of the model on unseen data.

Each sample $(\boldsymbol{x}, y)$ represents one flight, which input features $\boldsymbol{x}$ (see Table I) were extracted from: (1) the corresponding EFD message, (2) the last METAR available before the look-ahead time, and (3) traffic demand computed from the whole collection of EFD messages. Obviously, the target y is the identifier of the SID (or STAR) executed by that flight.

The total number of unique SIDs executed at EHAM during 2019 is 23, which frequency distribution is very imbalanced: the most frequent SID was executed by 14K flights, while the least frequent SID by only 13. Analogously, the number of unique STARs executed at EHAM during 2019 is 11, and the frequency distribution is even more imbalanced than for the SIDs: the most frequent STAR was executed by 20K flights, while the least frequent STAR by only 500 flights.

Dealing with classification problems where the classes are not represented equally is a dangerous situation that could lead to the well known *accuracy paradox*, in which the accuracy of the model is supposed to be excellent, but in reality is only reflecting the underlying class distribution. In this paper, the loss function Eq. (1) was modified to assign a weight to each sample $(\boldsymbol{x}, y)$, inversely proportional to the frequency of y:

TABLE II: Optimal *hyper*-parameters of the models

Hyper-parameter	Procedure	
	SID	STAR
`lambda_l1`	6.7e-6	0.
`lambda_l2`	1.68	0.
`num_leaves`	304	56
`feature_fraction`	0.52	0.68
`bagging_fraction`	0.47	1.
`min_child_samples`	44	20

$$w_i = \frac{|\mathcal{T}|}{|\mathcal{Y}||\{(\boldsymbol{x}, y) \in \mathcal{T} : y = i\}|}, \quad i \in \mathcal{Y}. \tag{4}$$

Accordingly, a higher penalty is given to the model when it miss-classifies a minority (i.e., infrequent) class.

The selection of correct values for the *hyper*-parameters of the model could boost its performance. In this paper, the tree-structured parzen estimator (TPE) implemented in the *optuna* framework[3] [18] has been used to optimise the *hyper*-parameters of the GBDT. Roughly speaking, TPE is Bayesian optimisation method which algorithm selects the *hyper*-parameters and corresponding values to evaluate in the next iteration based on the distribution of the previous results.

Many *hyper*-parameters can be used to optimise a GBDT, which allow to control the entire ensemble as well individual decision trees. Table II shows the optimal *hyper*-parameters of the models, resulting from the TPE optimisation.

`lambda_l1` and `lambda_l2` are regularisation terms, which limit the magnitude of model weights by adding a penalty in order to control over-fitting. num_leaves is used to limit the maximum number of leaves of each trained tree. Typically, setting a large `num_leaves` improves the performance on the training set, but also increases the chances of over-fitting. `feature_fraction` and `bagging_fraction` parameters are used to randomly select a subset of features and data, respectively, on each iteration (i.e., when fitting a new decision tree). Finally, `min_child_samples` determines the minimum number of samples required in a leaf. See the official documentation of lightGBM[4] for further details.

B. Illustrative example

Figure 2 shows an illustrative example of SID prediction for a flight in the test set. This figure shows the probability (in percentage) associated to each SID, as predicted by the model 4 hours before take-off. Note that all probabilities sum up to 100%. The blue-solid line in Fig. 2 is the actual (i.e., executed) SID, and the grey-dashed line represents the expected SID according to the EFD when performing the prediction.

This particular example corresponds to a flight from EHAM to Seville (LEZL), operated by Transavia (TRA) in November 2019. According to the latest METAR available when performing the prediction, the wind was blowing from NE. Furthermore, the expected SID in the EFD was LARAS.

[3]https://optuna.readthedocs.io
[4]https://lightgbm.readthedocs.io

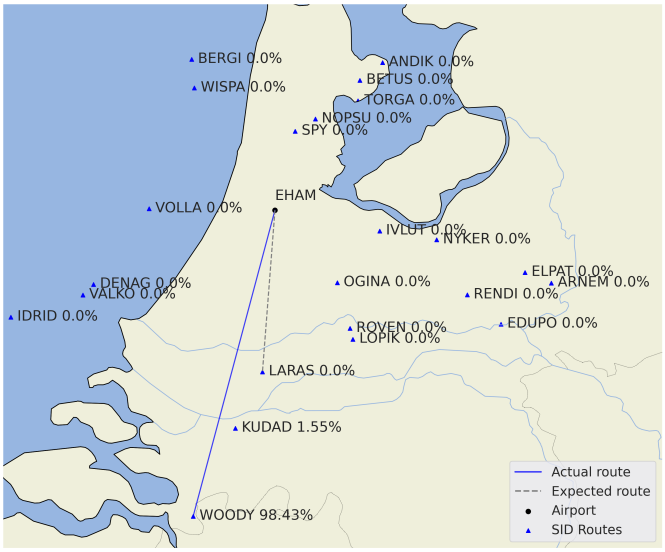


Figure 2: Illustrative example of SID prediction – EHAM to LEZL operated by TRA during November 2019, when the wind was blowing from NE

According to the model, however, the probability of executing LARAS was infinitesimal. Given the specific conditions of that flight (captured in $\boldsymbol{x}$), the model predicted with high confidence that the executed SID would be WOODY (98.4%). KUDAD was the most likely SID after WOODY, albeit with a significantly lesser probability (1.55%). Actually, WOODY was the SID executed by that flight, thus demonstrating a convincing judgement by the model well before take-off.

As expected, the probability associated to the SIDs which last point is located in the opposite direction to LEZL (which is located in the south-west of EHAM) was insignificant.

This instructive example depicts an intriguing case in which the model estimates a small likelihood for the expected SID in the EFD. If an individual does not know why such drastic statement is made, he or she may question the credibility of the prediction. In this paper, dedicated metrics and visualisation techniques based on feature attribution methods have been used to explain *why* a model performs a specific prediction given the inputs, aiming to increase the level of trust.

Figure 3 shows the *force* plot (more details can be found in [19]) for the actual SID (a), the expected SID (b) and a SID which last point is in the opposite direction to LEZL (c).

In this kind of plot, red arrows represent features that drive the prediction to higher probabilities, while blue arrows are those features that drive the prediction to lower probabilities. The size of each arrow represents the magnitude of the importance $\phi_i(\boldsymbol{x})$ for the corresponding feature i. The output value $f(\boldsymbol{x}, \Theta)$ corresponds to the logarithm of the odds predicted by the model, and the base value represents the expected value (i.e., the prediction of a hypothetical feature-less model). The higher the value of $f(\boldsymbol{x}, \Theta)$, the more probability is associated to the corresponding class (i.e., SID). Note that only the features with the most important effects are shown.

TABLE III: Aggregated performance metrics on the test set

Procedure	Model	Macro average (weighted average)			
		Precision	Recall	F1-score	Accuracy
SID	Baseline	0.40 (0.76)	0.31 (0.23)	0.17 (0.30)	0.23
	GBDT	0.85 (0.93)	0.73 (0.93)	0.76 (0.93)	0.93
STAR	Baseline	0.76 (0.85)	0.55 (0.72)	0.51 (0.69)	0.72
	GBDT	0.81 (0.87)	0.77 (0.87)	0.78 (0.87)	0.87

Figure 3a shows that most of the input features act to increase the probability of executing WOODY, and those that do not have marginal effects. Interestingly, the expected SID (LARAS) also increases the probability of executing WOODY. Note that the destination airport (LEZL) and the wind direction (NE) have a significant contribution on WOODY's likelihood.

On the other hand, Fig. 3b shows that, while some of the features like the expected SID, the aircraft operator, the arrival traffic demand or the month of the year suggest that the expected SID (LARAS) will be executed, the destination airport and the wind direction (among other features with lower importance) strongly propose that it will not.

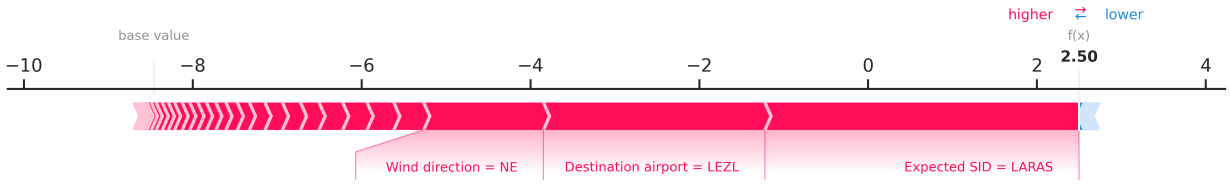
Finally, Fig. 3c shows that, despite some features increase the probability of a SID in the opposite direction to the destination airport, their combined effect is residual if compared to that of the features that reduce the probability. As expected, the destination airport notably decreases the likelihood of executing a SID in the opposite direction.

C. Aggregated performance metrics

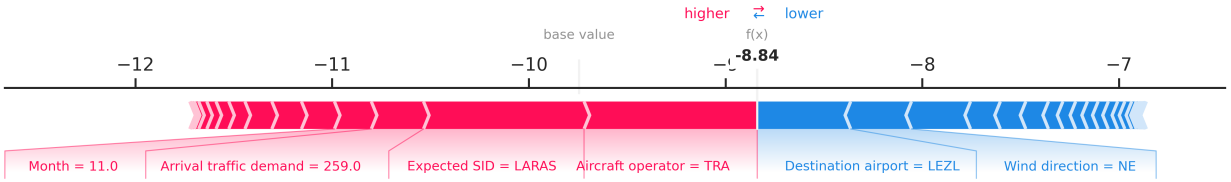
Table III shows the performance of the SID and STAR models, evaluated on the samples of the test set. In addition, the performance of a baseline model that takes the expected procedure as reported in the EFD as the best information is also shown for comparison purposes. For each model, Table III shows the most common metrics used to assess the performance of classification models [20].

In a point of fact, these metrics could be obtained for each model and each individual class (e.g., the precision of the GBDT model when predicting the class WOODY). Illustrating and interpreting the results using this granularity, however, would not be practical. For this reason, Table III shows, for each combination of model and metric, the average of all classes. In this analysis, two different strategies have been used to compute the average: macro average simply consists of summing the metrics of all classes, and then dividing by the number of classes. As such, the macro average highlights when the model does not perform well with the minority classes; as its name indicates, in a weighted average the average is performed by weighting the metric of each class according to its frequency, i.e., those with more samples will be favoured.

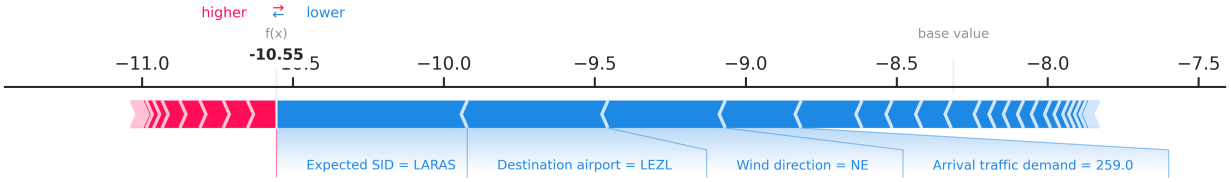
According to Table III, the precision of the SID model is 0.85 (0.93) 4 hours before ETOT, outperforming the precision of the baseline in the same conditions: 0.40 (0.76). Essentially, the precision answers the question to: Of all SIDs that the model predicted, how many were actually executed?".



(a) Actual SID (WOODY)



(b) Expected SID (LARAS)



(c) SID in the opposite direction (WISPA)

Figure 3: Force of the features for the illustrative example shown in Figure2

Similar conclusions apply to the recall, which answers the question to: ”Of all the SIDs that were actually executed, how many did the model identify?”. The SID model improves the recall of the baseline from 0.31 (0.23) to 0.73 (0.93).

Finally, F1-score is the weighted average (i.e., harmonic mean) of precision and recall such that the lowest value is highlighted. In other words, if the precision or the recall is small, the other metric no longer matters. According to Table III, the F1-score of the SID model is 0.76 (0.93), much higher than that of the corresponding baseline: 0.17 (0.30).

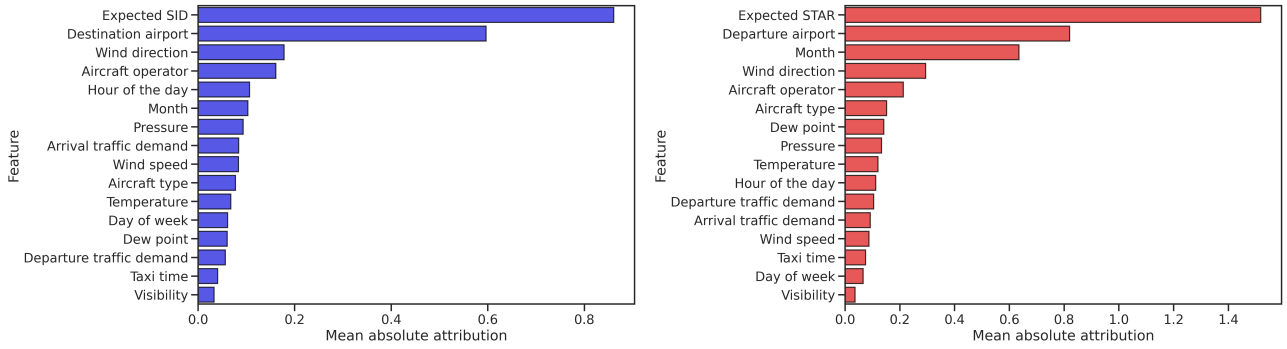
For the STAR model, the benefit with respect to the baseline is more modest than for the SID, yet the improvement is still notable. For instance, the STAR model increases the F1-score of the baseline from 0.51 (0.69) to 0.78 (0.87).

D. Model explainability

Figure 4 shows the mean absolute attribution (i.e., the importance) of each feature considering all samples in the test set. Note that features related to the traffic demand (arrival and departure) of each individual SID or STAR (see Table I) have been removed from the figure for the sake of clarity.

Results show that both SID and STAR models give a lot importance to the expected procedure as well as to the destination and departure airports, respectively. Interestingly, for the SID model, the direction of the wind is the 3rd most important factor, but for the STAR model, it is the month of the year. The aircraft operator also appears in the top of the ranking for the two models, suggesting that the executed SID or STAR heavily depends on who is operating the flight. Last but not least, the importance of features related to the traffic load (arrival and departure traffic demand) is relatively small if compared to more specific features like the wind direction or the aircraft operator, which suggest that a simpler version of the models using only flight, weather and calendar features may provide equivalent performance and notably simplify the feature construction process.

Note that the most important features in Figure 4 coincide with those driving the prediction for the illustrative example. Furthermore, this figure aims at summarising, in a simple way, the importance of each feature individually. The interaction between the features, however, is very complex, and different combinations of features may result in totally distinct results.



(a) SID model

(b) STAR model

Figure 4: Features importance

V. Conclusions

This paper proposed a generic gradient boosted decision trees (GBDT) model to predict standard instrumental departures (SID) and standard terminal arrival routes (STAR). The GBDT model makes use of features related to the flight for which the route is predicted, weather, calendar and traffic load at the airport to compute the most likely SID and STAR 4 and 5 hours before take-off and landing, respectively.

Promising results from a case study for Amsterdam-Schiphol airport using traffic and weather data from 2019 showed that the model outperforms the predictions of a baseline that uses the expected procedure collected from the enhanced tactical flow management system flight data messages as the best information. This simple model could help to improve demand predictions, as well as to predict the future noise levels in the communities near the airport.

In future work, the model must be evaluated in other major airports. Furthermore, the interaction between the input features must be analysed in order to improve the model explainability and build more trust on the model. In this context, a simplified variant model using only the most important features could be also investigated. A comparison with other models, such as artificial neural networks, is also foreseen.

Last but not least, the model has been trained to perform predictions at specific look-ahead times. In future work the model must be enhanced to perform predictions at any time, in order to investigate whether it experiences the same drop in performance as the hours before take off / landing increase.

References

[1] F. Rehm, "Clustering of flight tracks," in *Infotech@Aerospace*, 2010.

[2] M. Enriquez, "Identifying Temporally Persistent Flows in the Terminal Airspace via Spectral Clustering," in *10th USA/Europe Air Traffic Management Research and Development Seminar*, Chicago, IL, 2013.

[3] M. C. R. Murça, R. J. Hansman, L. Li, and P. Ren, "Flight trajectory data analytics for characterization of air traffic flows: A comparative analysis of terminal area operations between new york, hong kong and sao paulo," vol. 97, 2018, pp. 324–347.

[4] X. Olive and J. Morio, "Trajectory clustering of air traffic flows around airports," *Aerospace Science and Technology*, vol. 84, pp. 776–781, 2019.

[5] S. J. Corrado, T. G. Puranik, O. J. Pinon, and D. N. Mavris, "Trajectory clustering within the terminal airspace utilizing a weighted distance function," *Proceedings*, vol. 59, no. 1, 2020.

[6] X. Olive, L. Basora, B. Viry, and R. Alligier, "Deep Trajectory Clustering with Autoencoders," in *9th International Conference for Research in Air Transportation (ICRAT)*, Tampa, FL, 2020.

[7] R. Marcos, O. García-Cantú, and R. Herranz, "A machine learning approach to air traffic route choice modelling," in *8th SESAR Innovation Days*, Salzburg, Austria, December 2018.

[8] Q. Duong, T. Tran, D.-T. Pham, and A. Mai, "A simplified framework for air route clustering based on ads-b data," in *International Conference on Computing and Communication Technologies*, 2019.

[9] H. Naessens, T. Philip, M. Piatek, K. Schippers, and R. Parys, "Predicting flight routes with a Deep Neural Network in the operational Air Traffic Flow and Capacity Management system," EUROCONTROL Maastricht Upper Area Control Centre, Maastricht Airport, The Netherlands, Tech. Rep., December 2017.

[10] Y. Liu and M. Hansen, "Predicting aircraft trajectories: A deep generative convolutional recurrent neural networks approach," 2018.

[11] M. C. Rocha Murça and M. de Oliveira, "A data-driven probabilistic trajectory model for predicting and simulating terminal airspace operations," in *39th Digital Avionics Systems Conference (DASC)*, 2020.

[12] J. Zhang, D. Mucs, U. Norinder, and F. Svensson, "Lightgbm:an effective and scalable algorithm for prediction of chemical toxicity–application to the tox21 and mutagenicity datasets," *Journal of Chemical Information and Modeling*, vol. 59, no. 10, pp. 4150–4158, 2019.

[13] Hans Koolen and Ioana Coliban, *Flight Progress Messages Document*, EUROCONTROL, Brussels, Belgium, 2019, edition No. : 2.501.

[14] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *22nd International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[15] 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," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 3146–3154.

[16] R. Dalmau, F. Ballerini, H. Naessens, S. Belkoura, and S. Wangnick, "An Explainable Machine Learning Approach to Improve Take-off Time Predictions," *Journal of Air Transport Management*, vol. 95, 2021.

[17] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017, pp. 4765–4774.

[18] A. Takuya, S. Shotaro, Y. Toshihiko, O. Takeru, and K. Masanori, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *25th International Conference on Knowledge Discovery & Data Mining*, 2019.

[19] S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K.-W. Low, S.-F. Newman, J. Kim *et al.*, "Explainable machine-learning predictions for the prevention of hypoxaemia during surgery," *Nature Biomedical Engineering*, vol. 2, no. 10, p. 749, 2018.

[20] A. Geron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017.