A Deep Learning Approach To Predict General Aviation Traffic Counts

Amir Abecassis, Daniel Delahaye, Moshe Idan

10.2514/6.2024-2701

A Deep Learning Approach To Predict General Aviation Traffic Counts

Amir Abecassis, OPTIM team, amir.abecassis@enac.fr
French Civil Aviation University (ENAC) lab, Toulouse, France

Daniel Delahaye, OPTIM team, daniel.delahaye@enac.fr
French Civil Aviation University (ENAC) lab, Toulouse, France

Moshe Idan, Faculty of Aerospace Engineering, moshe.idan@technion.ac.il
Technion - Israel Institute of Technology, Haifa, Israel

General Aviation traffic prediction is a major concern for Air Navigation Service Providers with a direct impact on air traffic flow and capacity management measures. This paper introduces a Deep Learning methodology using meteorological and calendar data to predict General Aviation traffic. The methodology is evaluated in great detail using historical data from the Nice Cote D'Azur Terminal Control Center sectors with an increase of the global prediction performance of 32% with Recurrent Neural networks-based models compared to current tools used in operation. Additional tools are finally proposed to analyze and attain an in-depth understanding of the predictions generated by the various models.

Nomenclature

X	=	Vector of features, entry of the model
y	=	Target of the model
D	=	The whole dataset used
D_{train}	=	Part of the dataset for model learning or training set
D_{test}	=	Part of the dataset for model evaluation or test set
f, h, g, l	=	A Machine Learning or a Deep Learning model
$\Delta, METAR\_loss, loss_{all}, loss_{last}, MAE$	=	Loss functions
$a^{[p]}$	=	Output of the neurons of a Neural Network layer p
$W^{[p]}, b^{[p]}$	=	Weights/bias of the neurons of a Neural Network layer p
σ	=	Standard deviation
R	=	Activation function of a Neural Network layer
S	=	A coalition of features
ϕ_j	=	The Shapley value of feature j
I_j	=	The SHAP importance of feature j

I. Introduction

A. Operational Problem

General Aviation (GA) traffic directly impacts air traffic flow and capacity management measures. Therefore, GA traffic prediction is a key issue for the Air Navigation Service Provider (ANSP). Many major airports have to handle both GA and Commercial Aviation. While the Commercial Aviation framework is clearly defined thanks to flight plans, GA operations can occur anytime when the airport is open. ANSP has to organize its services taking this major difference into account to guarantee safety, i.e., always having staff available to handle both types of traffic. In cases of unexpected events, mainly a dramatic increase in GA traffic, the only safe solution is sometimes to regulate commercial traffic. Consequently, the trade-off between ANSP's quality of service and performance becomes a key point. It is considered very fragile as only rudimentary tools exist to tackle General Aviation traffic prediction.

The services delivered by the DSNA (French Air Navigation Service) to GA flights include, at least, the Flight Information Service and Alerts, which are mandatory as long as GA flights contact the Air Traffic Service by radio in class E to class G airspace. For class C or D airspace (no class B in France), the Control Service is also mandatory for these flights. DSNA has set up SIVs (Secteurs d'Information de Vol - Flight Information Sectors) managed by Air Traffic Controllers (ATCo) located in airports to deliver these services to GA flights in France airspace, the level of which depends on the class of airspace in each SIV.

Nice Cote D'Azur Terminal Control Center (TCC) is managed by the Nice airport, which is one of the major French airports. It is currently composed of three SIVs, depicted in Fig. 1, that geographically divide a big part of the TCC airspace under Flight Level (FL) 195: SIV1 (grouping SIV1.1 and 1.2 up to FL175 and FL145, respectively), SIV2 (up to FL145), and SIV3 (grouping SIV3.1 and 3.2 up to FL115 and FL145, respectively). These SIVs can contain Control Traffic Regions (CTR) and Terminal Manoeuvring Area (TMA) sectors that manage operations to and from a specific aerodrome. The management of this intricate airspace scheme is further complicated by the changing nature of the traffic, significantly affected by the season, day of the week, weather, or special events. Consequently, specific controllers are dedicated entirely to GA flights and related services in the SIVs.

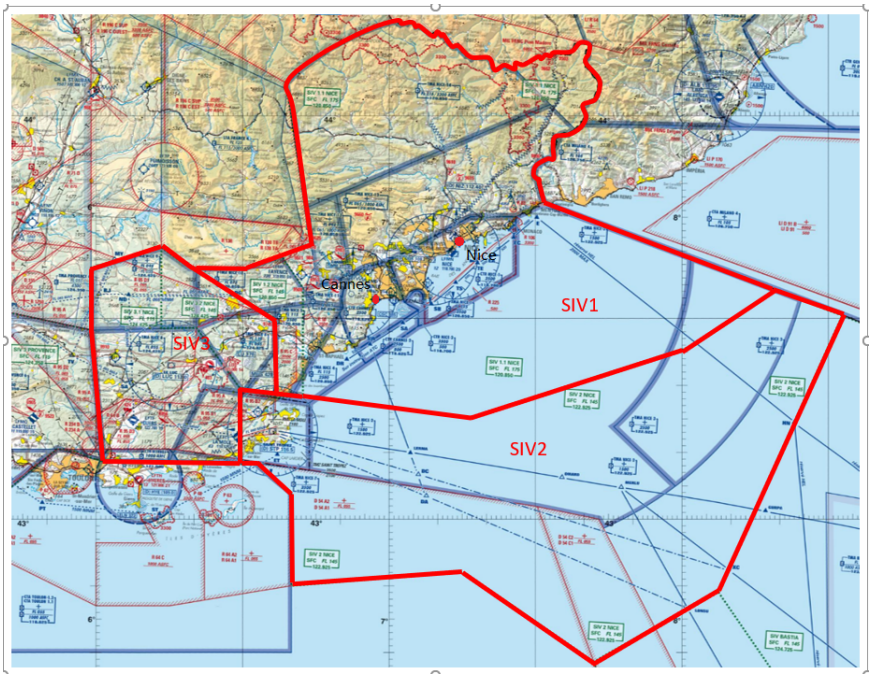


Fig. 1 SIV breakdown in the Nice TCC airspace.

B. The Initial Machine Learning Approach

Today, DSNA does not manage GA traffic globally, since it does not have the same impact on all its airports. However, as detailed above, the Nice TCC is one of the most impacted ones. Recently, a Machine Learning (ML) approach was proposed to mitigate this gap [1]. The purpose of the models presented in that study was to predict, four hours in advance, the GA flights entry counts in a defined period of one hour for a given air traffic sector. The approach is to cast the flight entry counts prediction as an optimization problem, solved using supervised ML methods.

In general terms, the goal is to predict a variable y (our target), in our case, the GA entry counts in a SIV, defined as an integer, from a set of explanatory features x, for which x and y are respectively drawn from probability distributions X and Y. Considering m features, $x \in \mathbb{R}^m$ and $y \in \mathbb{R}$. The random variable X is assumed to have a probability density function (pdf) $p_X(x)$. Our data is defined by the dataset D, with $D = \{(x_1, y_1), ..., (x_N, y_N)\} \subseteq (X, Y)$. The data in D are assumed to be independent and identically distributed (i.i.d). Consequently, the problem can be addressed by a supervised ML regression technique, whose goal is to approximate a function f such that: $y = f(x)\ \forall\ (x, y) \in (X, Y)$. Here, $f(x)$ is the prediction of our model, and y is the true value. The goal of this model, even if trained on a part of D, is to generalize well on an unseen dataset denoted by D_{test}, the data of which are also assumed to be i.i.d. and sampled

from the same (X, Y) distribution.

As with any supervised ML method, the performance of the methodology proposed in [1] strongly depends on the features engineering phase which constructs the sets of explanatory features x and target y. Calendar, meteorological, and GA flights entry counts historical data was explored to extract relevant features to identify the pilots' intent patterns. The feature extraction methodology adopted in [1] is detailed in the following paragraph. Although, as described next, the training set assembles historical data of three years, the features x include accurately only data available to the ATCos four hours before the target to predict. Nice TCC surveillance data were used to extract GA entry counts in each SIV, our y prediction target, considering the number of flights in a period of one hour for a total of 24 time slots per day. Figure 2 presents an example of counts on August 22, 2021.

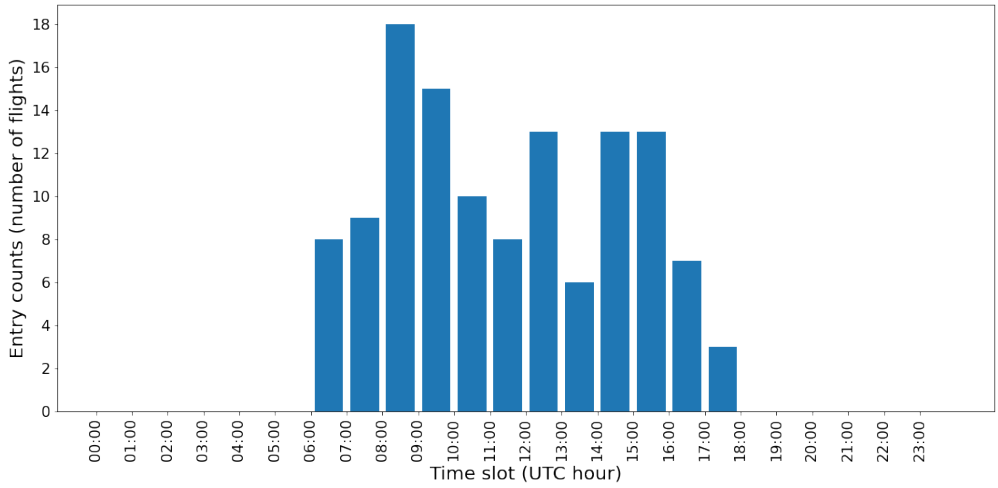


Fig. 2 Entry counts for SIV1 on Sunday, 22th of August, 2021.

The first set of features used in x are meteorological data. Those are composed of METAR (Meteorological Aerodrome Report) and TAF (Terminal Aerodrome Forecast), which are, respectively, meteorological observation and forecast data in an aerodrome. Since the historical METAR and TAF data are available in free text format standardized by the International Civil Aviation Organization, a key step was the definition of a scaling scheme using operational personnel's expertise (controllers, pilots). This scheme is defined to capture the impact of meteorology on air traffic operations, producing a 1-5 rating (higher values imply more critical weather) while using the various keywords inside METAR and TAF messages. A complete table of those ratings per keyword can be found in [1]. For brevity, several of those are presented in Table 1. Each keyword is individually rated and the global METAR (or TAF) evaluation considers the maximal rate from all message keywords. Nice and Cannes aerodrome meteorological stations were chosen for SIV1 predictions providing the best compromise with regard to the sector geography.

Table 1 METAR and TAF keywords evaluation examples.

Keyword	Type	Rating
FEW	Clouds	1
FU / DZ / MIFG	Significant weather	2
$\geq$ G20 and $<$ G30	Gust	3
M10	Temperature	4
PO / TS / SHRA / SHSN / GR / PL	Significant weather	5
"1500"	Visibility	5

The second set of features used are calendar data that includes, for each day/slot, the week number in the year (WN), the day number in the week (DoW), the time slot of the day (H, means the 1-hour slot duration from H to H+1). Note that since WN is related to the week number in a year, W1 and W53, that are separated greatly numerically are close

from a time and seasonal perspective. To properly capture this periodicity, the WN feature is converted into two features using cyclical features encoding [2], i.e.,

$$\mathrm{WN}^{\cos} = \cos\left(\frac{2\pi \cdot \mathrm{WN}}{53}\right), \tag{1a}$$

$$\mathrm{WN}^{\sin} = \sin\left(\frac{2\pi \cdot \mathrm{WN}}{53}\right). \tag{1b}$$

Finally, information about whether the day is a bank holiday or is during school holidays (S_{hol}), and information about whether a specific event is organized during the day (S_{ev}) were also added through two Boolean features. For the last feature, a list of events impacting regional GA traffic according to the ATCo's experience was identified. A sample of such features for 2019 is presented in Table 2.

Table 2 Some special events around Nice in 2019.

Event name	Start date in 2019	End date in 2019
Monte Carlo Rallye	24th of Jan	27th of Jan
Nice Carnival	16th of Feb	2nd of March
Cannes Festival	14th of May	25th of May
Monaco GP	26th of May	26th of May

In summary, the list of features used in [1] is presented in Table 3.

Table 3 List of Features used in [1].

- Nice_globalMET$_{H\text{-}4}$: Nice past METAR.
- Cannes_globalMET$_{H\text{-}4}$: Cannes past METAR.
- Nice_TAF$_H$: Nice TAF.
- Cannes_TAF$_H$: Cannes TAF.
- Nice_TAF$_J$: Nice average day TAF.
- Cannes_TAF$_J$: Cannes average day TAF.
- H : Time slot.
- DoW : Day of the week.
- WN^{cos} : Number of the week cosine.
- WN^{sin} : Number of the week sine.
- S_{hol} : Holiday.
- S_{ev} : Special event.

The historical data scope used in this study was limited to the period of January 2019 to May 2022. Due to COVID-19's impact on GA traffic, it was decided to remove the 2020 period. Random Forest (RF), Support Vector Machines, and Linear Regressor models were trained on the 2019/2021 data and evaluated on the 2022 data. The training and evaluation scope was limited at this stage to Nice TCC SIV1. The results were compared to the current tool in operation locally (mentioned as the Nice baseline in the present article) and demonstrated a 25% increase in the prediction performance.

Nice baseline predicts the entry counts (EC) of a given slot H by averaging the data available for the same time slot of the same day of the week, the same week number in the previous years. If EC is considered a function of H, DoW, WN, and year (YN), denoted in short by EC(YN), and considering available data excluding 2020 due to COVID-19, the entry count for a time slot in 2022 is predicted as

$$\mathrm{EC}(2022) = \frac{1}{2}\left[\mathrm{EC}(2021) + \mathrm{EC}(2019)\right]. \tag{2}$$

The approach presented in [1] and enhanced in the current study addresses also the over-fitting of the training part of D by balancing prediction accuracy (bias) and prediction error variance [3]. It will also affect the training methodology, as is addressed in the following sections.

The current paper extends this ML-based approach for predicting the GA entry counts aiming to improve the accuracy of the previous publication results using the same historical data scope for training and evaluation. The following improvements are considered: a new features engineering phase presented in section II, a pre-processing of the target to predict addressed in section II.D, and training of more advanced ML algorithms detailed in section III.A. Finally, section IV presents the results and analysis of these new models, proposing a state-of-the-art methodology based on Deep Learning with a wider evaluation of the three Nice SIVs. Concluding remarks are included in section V. Please refer to [1] for a complete review of the literature on ML applied to Air Traffic Management tasks, specifically for trajectory prediction, Demand and Capacity Balancing, and General Aviation prediction.

II. New Features Engineering Phase

Motivated by the results in [1], it was recognized that the performance of the proposed ML method can be improved by judiciously updating and pre-processing the training features of the method. Moreover, additional ML models are also considered here. This section presents the enhanced feature engineering phase that was guided by the statistical analysis of historical data.

A. Calendar Data

Results from [1] showed that a fine re-engineering of calendar data features, with a focus on the school holidays, bank holidays, and specific events impacts (represented previously by S_{hol} and S_{ev} boolean features) on traffic was necessary to bring more information to the model. To reach this objective, a statistical analysis has been performed on the training set to evaluate better the GA entry counts during the day of these specific events. It shows that the impact of the holidays on traffic is not the same depending on the period (e.g., during the Christmas holidays compared to the winter holidays) and that initial ATCo feeling about the impact of specific events on traffic was not aligned with reality (e.g., some initially selected events had negligible impact on GA traffic).

To better represent the effect of special events on the GA counts, we analyzed the difference between average day GA counts during specific events and during the entire training data period for all SIV sectors. The result of this analysis is presented in Figure 3. Consequently, a scale that represents the impact of specific events on GA traffic is proposed, where a higher value means a stronger effect on GA traffic. The chosen scale, summarized in Table 4, will constitute a new and single feature, called $S_{finalev}$. It will replace the former two Boolean features S_{hol} and S_{ev}.

Table 4 New specific event feature.

Event name	Impact
Monte Carlo Rallye / Ascension weekend	3
Nice Carnival / Cannes Festival / Monaco GP	2
Yacht Festival Cannes / Regates Royales Cannes / Voiles d'Antibes	1
Christmas holidays	-1
Other event	0

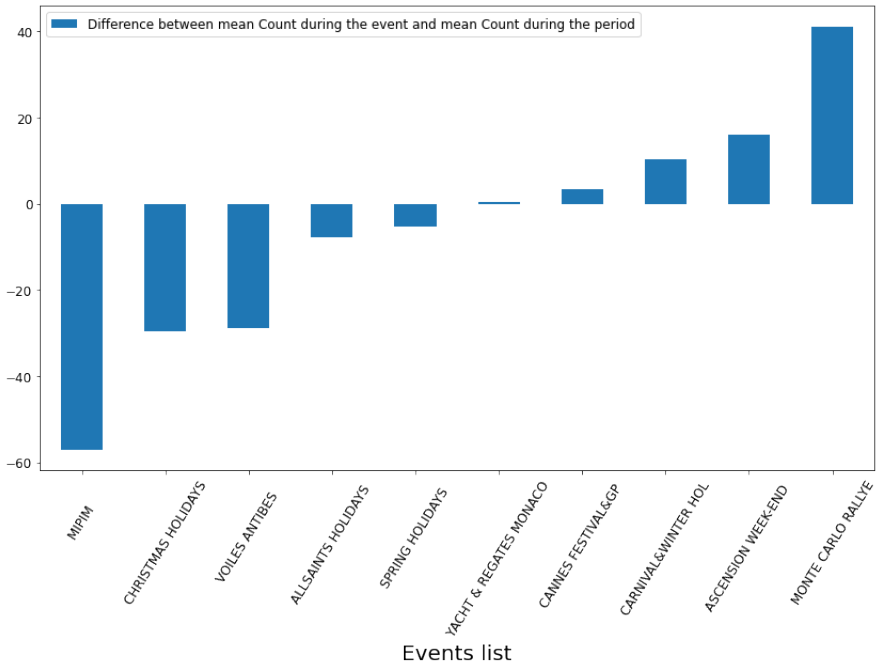


Fig. 3 Difference of GA average day counts during specific events & corresponding period on SIV1,2 and 3 during 2019/2021.

B. Meteorological Data

Considering the meteorological features, this part proposes two aspects of improvement compared to the baseline work performed in [1]. The first aspect relates to better use of the METAR and TAF data utilized before, while the second is aimed at identifying a new feature extracted from an additional data source, specifically the meteorology radar information.

1. A Better Use of METAR and TAF Data

METAR and TAF messages from different aerodromes were selected in correlation with their geographic location in the sector of interest. As this article focuses on other Nice TCC SIVs, other meteorological stations were selected considering the geographic correlation of SIV2 and SIV3: Toulon (LFTH) and Bastia (LFKB) were added to the initial selection of Nice (LFMN) and Cannes (LFMD). Data was extracted from historical METAR of Iowa Environmental Mesonet from Iowa State University [4] and OGIMET [5] to construct the same features as before for those new stations (i.e., xxxx_globalMET$_{H\text{-}4}$, xxxx_TAF$_H$, xxxx_TAF$_J$ - see Table 3 for the naming conventions).

In [1] it was concluded that to consider the meteorological scale built and presented in Table 1 was probably too simple to fully capture the complexity of the meteorological impact on the traffic. Nonetheless, this selection allowed us to attain initial models and results, i.e. error distribution bias and variance that are expected to be improved. For instance, the existing meteorological scale did not really focus on the temperature impact on traffic apart from extreme temperature conditions. The purpose of the following modifications was to identify new relevant features to enhance the initial selection.

First, the set of features extracted from METAR was expanded to include also: three temperature features (Air Temperature $tmpf$, Apparent Temperature $feel$ - Wind Chill or Heat Index, Dew Point Temperature $dwpf$), Relative Humidity $relh$, Pressure altimeter $alti$ and Visibility $vsby$. Figure 4 illustrates the correlation of the features with the traffic count focusing on Nice meteorological features. For comparison, also the calendar features are included in this chart. It appears clearly that the new selection has a correlation in absolute value between 0.1 and 0.2 with traffic count on the training set. At the end, these six new features were added for each of the four relevant meteorological stations cited previously: $Nice\_tmpf_{\text{H-4}}$, $Nice\_dwpf_{\text{H-4}}$, $Nice\_relh_{\text{H-4}}$, $Nice\_alti_{\text{H-4}}$, $Nice\_vsby_{\text{H-4}}$, $Nice\_feel_{\text{H-4}}$ are the features notation for Nice.

Then, instead of only using the conversion of the whole meteorological information included in METAR directly to

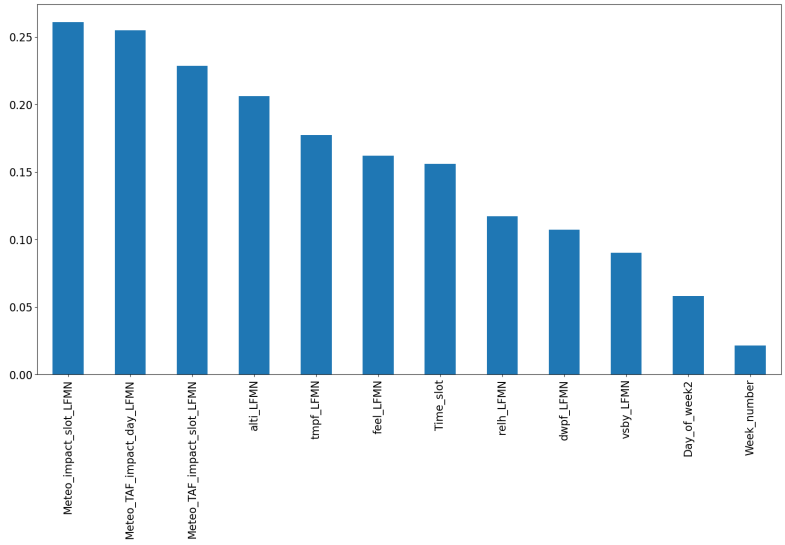


Fig. 4 Correlation between features and traffic count on the training set in absolute value.

this synthetic scale, a specific features extraction phase based on a Deep Learning approach is proposed. It consists in training a dedicated model (aka, a neural network) to learn the most relevant and low-dimensional representation of METAR to predict the meteorological scale level considered as 5 categories.

The solution is based on tokenization and embedding layers that convert each word of the METAR message into positive integers and then into dense vectors. This idea was inspired by solutions used in Natural Language Processing for word representations with numeric versions like Word2Vec [6]. Then, embedding layers outputs feed an Autoencoder Neural Network [3, Ch. 14]. This type of Neural Network (NN) is trained to learn its own input in an unsupervised manner. Its architecture, presented in Fig. 5, is composed of two components: an encoder and a decoder, denoted, respectively, as functions g and h. The encoder squeezes the NN input through a bottleneck layer (with fewer nodes than the input layer) into a lower-dimensional representation (the code). Then the decoder tries to reconstruct the input from this representation. Controlling the code dimension allows to learn the best input representation in this dimension. In other words, the compression allows to minimize the loss of information to reconstruct the input. From a mathematical point of view, the problem is to learn g and h functions that satisfy [7]

$$\min_{g,h} \mathbb{E}\left[\Delta\left(X, h \circ g(X)\right)\right] \tag{3}$$

with X the probability distributions of the data and Δ the reconstruction loss function (often l_2 norm).

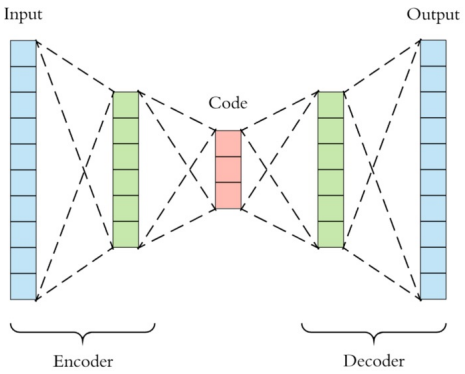


Fig. 5 Presentation of an autoencoder architecture.

The tokenization process only keeps 2800 most cited words (others are set to Unknown token). Each METAR message is considered as a sequence of 25 tokens (if necessary METAR message are padded or truncated to fit to this

size). The embedding layer is set to a dimension of 8. To better capture the information provided by a METAR sequence of words, the architecture of the Autoencoder is modified using recurrent layers instead of Dense fully connected layers [8]. Long Short Term Memory (LSTM) with hyperbolic tangent activation functions have been selected because they are more robust than simple RNN layers to tackle long sequences and to better address the vanishing gradient effect [3]. The number of neurons of the LSTM layer is set to 8 for both the encoder and decoder. LSTM encoder output feeds the decoder composed of an LSTM layer (same number of neurons, same activation function) and then a Dense Fully connected layer (Dense) with a normalized exponential function (softmax) as activation function.

In parallel, the encoder output (the code) also feeds a classic Artificial Neural Network (ANN) represented by function l. Its role is to predict the meteorological scale level (1 to 5) based on the code representation. This second Neural Network, trained in parallel with the previous one, helps the first one to learn the best representation useful to capture the impact on the GA traffic. The ANN is composed of 2 Dense layers with, respectively, 100 and 50 neurons and a rectified linear unit (ReLU) activation function. The output layer is a Dense layer with 5 neurons and a softmax activation function.

The whole network (METAR_network), composed of the Autoencoder and the ANN, is trained to optimize a customized loss function ($METAR\_loss$). If y_true is the true target category for meteorological scale impact on GA traffic (both of 5 dimensions) and x is the METAR sequence representation (25 dimensions), the loss function is defined by

$$METAR\_loss = \left[\sum_{i=1}^{5} y\_true_i * \log\left(l \circ g(x_i)\right)\right]^2 + \left[\sum_{j=1}^{25} x_j * \log\left(h \circ g(x_j)\right)\right]^2 . \tag{4}$$

The global architecture for meteorological data features extraction is presented in Fig. 6. The METAR_network was trained on a set of around 450000 LFTH, LFKB, LFMN, LFMD METAR from 2014 to 2018 with an Adam optimizer [9] and specific hyper-parameters (learning rate: 0.001, epochs: 100, batch size: 32, exponential decay rate for the first moment estimates: 0.9, exponential decay rate for the second moment estimates: 0.999). When trained, the model allows to extract from each METAR an 8-dimension representation using the encoder part of Autoencoder. This brings more information than the basic meteorological scale used in [1] and is considered as additional features (e.g., $Nice\_MET\_feat1_{\mathrm{H4}}$ is feature 1 or first dimension of 8-dimension vector output generated by encoder part of METAR_network for Nice station METAR available for slot $H-4$).

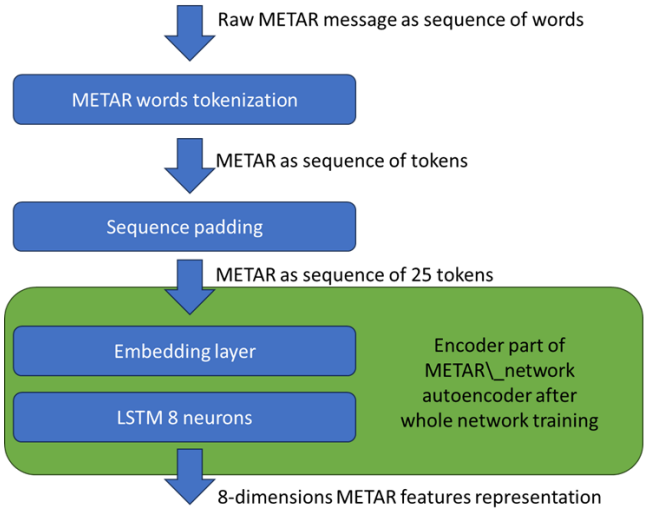


Fig. 6 METAR features extraction through the autoencoder pipeline.

2. New meteorological sources

METEO FRANCE ASPOC service allows to identify weather hazard objects that directly impact aviation, most of the time linked to convection [10]. Convection, in weather, describes the vertical transport of heat and moisture in

the atmosphere. When convection is severe, it is generally associated with thunderstorms. This real-time information is observed by ATCo using meteorology radars. It was possible to get METEO FRANCE ASPOC archives through MetSafe API [11] and to extract events that occurred in each of the three Nice SIVs. Each ASPOC message identifies a weather hazard, the 5-minute slot while it is observed and its severity (light / moderate / moderate severe / severe). The severity category is converted to a severity number from 1 (light) to 4 (severe) and the severity numbers of all the events on each 1-hour slot are summed (0 if no event occurred during the slot). The resulting number is the new $ASPOC\_intensity$ feature integrated into our features vector and the information used for the prediction is the $ASPOC\_intensity$ available 4 hours before the target slot to predict.

C. GA counts related to SIV2 and SIV3

The geographic location of SIV2 and SIV3 is very close as both are covering a major part of maritime traffic. So, even if they are currently separated in the Nice TCC for operational reasons, it was considered relevant to tackle both SIVs together. As a consequence, GA counts of both SIVs were merged into a specific sector, called SIV2/3 for the remainder of the paper.

D. Pre-Processing of the Target to Predict

One lesson learned from [1] is that the use of 1-hour time slots made the data jumpy, possibly affecting the performance of the model. To address this issue, we propose pre-processing the GA entry counts (ML targets) with a smoothing method called averaging windows applied before the ML model training. This processing intends to smooth the effect of considering a 1-hour slot, starting at a fixed time (e.g., from 8 AM to 9 AM). From a mathematical perspective, the processing relies on the following idea. Instead of taking the entry counts (EC) from the start of a 1-hour time slot to its end (for instance, from 8 AM to 9 AM), a window is slid over the slot every minute keeping the whole time slot as a constraint (so counting, for example, from 8:01 AM to 9:01 AM) to generate new counts. This is done over an interval of [-15 minutes, +15 minutes] around the initial slot. The new counts (avg_{EC}) obtained over each 1-hour time slot are the average of those intermediate counts. Specifically, if EC_i is the count obtained when the window is slid by i minutes, the average count is computed by

$$avg_{EC} = \frac{1}{31} \sum_{i=-15}^{15} EC_i. \tag{5}$$

This average avg_{EC} replaces the original target for ML model training.

III. Training of State-of-the-Art ML Models

A. New Models

Several types of models were considered: Gradient Boosting Decision Trees (GBDT), Artificial Neural Networks (ANN), ANN ensemble (ANN_ens), mixed Convolutional Neural Network and Recurrent Neural Networks (RNN), RNN with training on last the time step (RNN_last), and RNN with training on all the time steps (RNN_all) [3, 12]. These models were trained and their performance was compared with each other and with the Nice baseline of (2) and with the best former RF baseline [1].

GBDT is an ensemble model based on Decision Trees (DT) trained sequentially. Each new tree added is trained to optimize previous tree residual errors with gradient descent. However, the model selected in this article is an improved version of GBDT called LightGBM [13]. This version is known to optimize training speed and memory usage using histogram-based algorithms, accuracy using a leaf-wise strategy for DT growing (instead of a level-wise strategy), and the split for categorical features using histogram property. In addition, it uses the Goss method [13] which is the newest and the lightest implementation of GBDT. It suggests a training data sampling method based on gradient calculation to avoid searching on all the data space. However, in the present article, the sampling strategy was kept to a default bagging strategy considering the reasonable size of the dataset.

ANN model is a classic MultiLayer Perceptron Neural Network based on the composition of fully connected Dense layers with nonlinear activation functions. If $a^{[p]}$, $W^{[p]}$, $b^{[p]}$ represent the output, the weights/bias of the neurons of the layer p, R the activation function of the related layer (e.g. RELU / sigmoid function for instance), the relation

between each layer output is

$$a^{[p]}(x) = R\left(W^{[p]}.a^{[p-1]} + b^{[p]}\right). \tag{6}$$

The output of the model is the composition of the output of each of its layers. The model is trained by minimizing the loss of the network using a cost function over the training set that compares predictions with ground truth. The NN weights and bias are updated through the backpropagation technique [3]. ANN_ens is an ensemble model based on prediction averaging of the best ANN models trained during the hyper-parameters tuning phase [14].

RNN models are Recurrent Neural Networks using past data information. Considering the previous models, all the features provided to each of them were static. In this phase, the model is designed to analyze and extract the relevant features from the past (168 steps, which is equal to one week of past data information). RNN can be considered as an ANN that has been unfolded through time. The calculation of the loss is identical to ANN and a specific technique called backpropagation through time is used to calculate RNN bias and weights updates [3]. Several versions have been evaluated:

- RNN version is a RNN with Long-Short-Term-Memory (LSTM) or Gated Recurrent Unit (GRU) layers. These layers are known to be more robust than simple RNN layers to tackle long sequences and to fight against the vanishing gradient effect [3] by putting in place specific gates to manage the memory to keep from the past [15, 16];
- CNN/RNN version is the same RNN adding a preliminary feature extraction level based on 1D-convolutional layer processing (Convolutional Neural Network) [17]. This level addition is expected to deal with a very long sequence, reducing its size by convolution.

For RNN, two sub-versions are trained. First, RNN_last is trained by calculating the loss (MAE) only on the last time step of the sequence. If (x, y) is a set of (features, target), n the length of the sequence, f the RNN model, the corresponding loss evaluated on the last time step ($loss_{last}$) is

$$loss_{last}(x, y) = MAE\left(f(x_n), y\right). \tag{7}$$

Then, RNN_all is trained by calculating the loss (MAE) on all the time steps of the sequence. The corresponding loss evaluated on all-time steps ($loss_{all}$) is

$$loss_{all}(x, y) = \frac{1}{n}\sum_{i=1}^{n} MAE\left(f(x_i), y_i\right). \tag{8}$$

It is important to note that the RNN models allow to consider complementary features: GA flights counts of past slots (4 hours before the slot to predict) and all the already identified features information (calendar, meteorological data) from the past on a $lookback$ period. This $lookback$ period represents the past time window considered by the RNN model and is measured by the number of time steps in the time window (as a time step is a 1-hour slot, for instance, a $lookback$ of 10 means a past time window of 10 hours).

B. Updated List of Features and the Models Training Procedure

Following the updated features engineering and the models chosen for training discussed above, the final list of features is:

- H : Time slot
- DoW : Day of the week
- WN^{cos} : Number of the week cosine
- WN^{sin} : Number of the week sine
- S_{finalev} : Special event of specific day new evaluation
- $ASPOC\_intensity$: Global evaluation of ASPOC impact of the SIV
- Nice_globalMET$_{H\text{-}4}$: Nice past METAR global evaluation
- $Nice\_MET\_feat1_{\text{H-4}}$ to $Nice\_MET\_feat8_{\text{H-4}}$: 8 new features to describe Nice METAR content
- $Nice\_tmpf_{\text{H-4}}, Nice\_dwpf_{\text{H-4}}, Nice\_relh_{\text{H-4}}$: Air Temperature, Apparent Temperature, Dew Point Temperature for Nice station
- $Nice\_alti_{\text{H-4}}, Nice\_vsby_{\text{H-4}}, Nice\_feel_{\text{H-4}}$: Relative Humidity, Pressure altimeter and Visibility for Nice station
- Nice_TAF$_H$: Nice TAF global evaluation

- Nice_TAF$_J$: Nice average day TAF global evaluation
- $Count$: Count of past slots (only available in RNN_all model)

Note: All the Nice features exist also for other meteorological stations (Toulon, Bastia, Cannes). This means that the final list of features is made up of 74 features. For RNN, considering the past information and the additional *count* feature, the number of features is 75*$lookback$.

All the ML models presented in section III.A are based on Supervised Learning and have been implemented and trained on the training data (D_{train}) to perform the regression task of GA entry counts prediction. The optimization criterion remains the Mean Absolute Error (MAE) between predicted and true entry counts in the SIV loss function as in [1]. For each ML model trained, the same methodology has been followed and is summarized in Fig. 7.

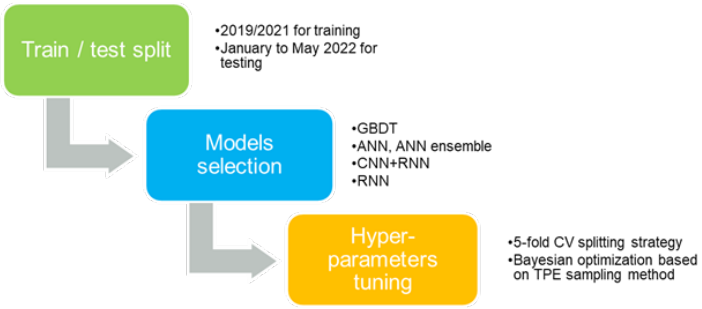


Fig. 7 Flowchart of data selection and model training.

The training methodology has been maintained as close as possible to [1] for results comparison in section IV. First, 2019 and 2021 historical data was used for D_{train}. 2020 data was removed from the training set due to the COVID period (lockdown and curfew impacts). Then, January to May 2022 data was used for D_{test}. This temporal split was done to avoid data leakage from the future to the past and to allow evaluation of the models on unseen data. Finally, a full day is represented by fifteen 1-hour time slots between 5 AM and 8 PM (Universal Time) as it was demonstrated that this period contains more than 99% of the daily traffic.

Additional parameters had to be set. For Neural Networks, the Adam stochastic optimization algorithm was used [9] with a batch size of 128 samples of D_{train} for gradient update calculation and update. An intermediate number of training period was set to 30 epochs for each Neural Network preliminary evaluation with early pruning of the model if its intermediate result was worse than the median of intermediate results for the already trained models. If not pruned, an additional training period was set to 130 epochs with an early stopping callback in case MAE stops improving on the evaluation dataset. For ANN, the Batch Normalization technique was systematically applied to reduce learning duration [18]. For RNN, as this solution is not efficient, the Layer Normalization alternative was applied [19]. A last RNN specificity is the definition of the $lookback$ period parameters that control the size of the time series and the past period used by the model. Based on the operational expertise, a $lookback$ of 105 time steps was chosen which is equivalent to one week of past data.

Hyper-parameters tuning strategy was a 5-fold cross-validation (CV) splitting strategy for the evaluation dataset [3] with a Bayesian optimization based on Tree Parzen Error sampling method [20]. For RNN and time series prediction, the classic 5-fold CV split is not appropriate as data splitting is done randomly, implying the possibility of using future values to forecast past values, i.e., data leakage. To prevent this, a stacked 5-fold CV splitting strategy was used for all the RNN-based model training. It consists of splitting the training set into five different subsets. The model is then trained five times in the following way: on training trial $i \in [1, \ldots, 5]$, the model is evaluated on the i-th subset considered as the present time data and training is done exclusively on the past data part of the dataset. Optuna framework was used for each model hyper-parameters tuning with a set of 100 trials to find the best results [21]. The set of all tunable model hyper-parameters that are used to control overfitting are [12]:

- n_estimators specifies the total number of DT that the model will learn;
- learning_rate determines the step size at each learning iteration while moving toward a minimum of a loss function;
- max_depth controls the maximum depth of each DT;
- $max\_samples$ defines the ratio from D to use for each DT training (bagging);
- $max\_features$ determines the number of features to consider when looking for the best DT split;
- $bootstrap$ controls if bootstrap samples are used when building DT [22];
- num_leaves controls the maximum number of tree leaves for each DT;

- reg_alpha, reg_lambda determine the L1 / L2 regularization terms on weights;
- min_child_samples defines the minimum number of data needed in a leaf child in a DT;
- subsample_freq, colsamp_tree specify the bagging frequency, the subsample features ratio for each DT construction;
- nb_hidden_dense_layers, nb_dense_neurons, activation_function_hl define the number of intermediate Dense layers, the number of neurons for each intermediate Dense layer, the activation function for each intermediate Dense layer (R) for CNN/RNN, RNN, ANN models;
- kernel_init, bias_init define the method for weights, bias matrix initialization;
- ker_reg_dense, bias_reg_dense, dropout_rate_dense control the regularization strategy on weights, bias and global dropout rate.
- beta_1, beta_2 control the exponential decay rate for the first moment estimates, the second moment estimates in Adam gradient-descent optimization algorithm [9];
- rec_layer_type, nb_hidden_rec_layers, nb_rec_neurons define the type of recurrent layers (LSTM, GRU), the number of recurrent layers, the number of neurons in each recurrent layers for CNN/RNN and RNN models. The default activation functions for these layers were used (hyperbolic tangent for the activation function / sigmoid activation function for the recurrent step) [15, 16];
- ker_reg_rnn, rec_reg_rnn, bias_reg_rnn, dropout_rate_rnn, rec_dropout_rate control the recurrent layers regularization by applying regularization strategies on the weights/bias matrices of each layer.
- cnn_filters defines the numbers of outputs filters in the convolution
- cnn_kernel_size, cnn_strides, cnn_padding specify the length of the 1D convolution window, the stride length of the convolution, the regularization strategy in the convolution layer for CNN/RNN models.

Table 5 presents the relevant hyper-parameters for each ML model. GBDT and NN have specific hyper-parameters for regularization: a L1/L2 regularization term can be applied on weights [23]. Dropout regularization techniques can be used in addition for the NN models [24].

Table 5 Tunable Hyper-parameters.

Model	Hyper-parameters list
RF	n_estimators, max_depth, $max\_samples$, $max\_features$, $bootstrap$
GBDT	n_estimators, max_depth, $max\_samples$, num_leaves, min_child_samples, subsample_freq, colsamp_tree, reg_alpha, reg_lambda, learning_rate
ANN	nb_hidden_dense_layers, nb_dense_neurons, activation_function_hl, kernel_init, bias_init, ker_reg_dense, bias_reg_dense, dropout_rate_dense, learning_rate,beta_1, beta_2
RNN*	rec_layer_type, nb_hidden_rec_layers, nb_rec_neurons, ker_reg_rnn, rec_reg_rnn, bias_reg_rnn, dropout_rate_rnn, rec_dropout_rate
CNN/RNN†	cnn_kernel_size, cnn_strides, cnn_padding

IV. Results and Analysis

The current paper expands the preliminary evaluation presented in [1], addressing all the methods and models presented above, on both Nice SIV sectors, SIV1 and SIV2/3.

A. Evaluation on Global Metrics

The first evaluation compares the following global metrics: MAE on D_{train} and on D_{test}, standard deviation (σ) and 95% Confidence Interval (CI95%) on D_{test}. The following families of models are evaluated:

- Models without averaging windows / without new features engineering phase;
- Models with averaging windows / without new features engineering phase (Without NFEP);
- Models with averaging windows / with new features engineering phase (With NFEP).

Table 6 and Table 7 sum up the evaluation of the models on these global metrics.

*RNN hyper-parameters also encompasses ANN ones.

†CNN/RNN hyper-parameters also encompasses RNN ones.

Table 6 Global results without averaging windows / without new features engineering phase.

Model	SIV1 sector			SIV2/3 sector		
	Train MAE	Test MAE / σ	$CI_{95\%}$	Train MAE	Test MAE / σ	$CI_{95\%}$
Nice baseline	NA	3.23 / 4.75	[3.08, 3.38]	NA	3.53 / 5.25	[3.36, 3.70]
RF baseline	2.07	2.45 / 3.58	[2.33, 2.56]	2.36	2.76 / 4.19	[2.62, 2.91]
GBDT	2.01	2.41 / 3.51	[2.30, 2.52]	-	-	-
ANN	1.98	2.54 / 7.35	-	-	-	-
ANN_ens	1.82	2.46 / 3.58	[2.34, 2.57]	-	-	-
CNN/RNN	2.21	2.54 / 7.36	[2.42, 2.66]	-	-	-
RNN_last	2.15	2.31 / 7.25	[2.21, 2.43]	-	-	-
RNN_all	2.03	2.30 / 3.35	[2.19, 2.41]	2.28	2.68 / 3.85	[2.54, 2.81]

Table 7 Global results with averaging windows / with and without new features engineering phase.

Model	SIV1 sector			SIV2/3 sector		
	Train MAE	Test MAE/σ	$CI_{95\%}$	Train MAE	Test MAE/σ	$CI_{95\%}$
Nice baseline	NA	3.15 / 4.63	[3.01, 3.30]	NA	3.46 / 5.11	[3.29, 3.62]
RF without NFEP	1.84	2.34 / 3.44	[2.24, 2.45]	2.25	2.64 / 4.02	[2.50, 2.78]
RNN_all without NFEP	1.70	2.22 / 3.26	[2.12, 2.33]	2.18	2.43 / 3.57	[2.31, 2.56]
RF with NFEP	1.67	2.25 / 3.26	[2.15, 2.35]	1.98	2.46 / 3.83	[2.33, 2.59]
RNN_all with NFEP	1.28	2.15 / 3.19	[2.05, 2.24]	1.78	2.38 / 3.59	[2.26, 2.50]

Considering only the improvements attained by using different models, addressed inTable 6 showing results without averaging windows and without new features engineering phase, the best baseline for the SIV1 sector is RNN_all performance criteria. It improves the Nice baseline by 29% and the previous state-of-the-art model (former RF) by 5% when evaluated using the test set of SIV1. This confirms that past data are relevant for model prediction. Moreover, it is important to note that RNN_all is the most robust as its standard deviation is the lowest which also yields the lowest 95% Confidence Interval. This evaluation also allows to conclude that other models evaluated do not add value compared to easy-to-train RF and best performing RNN_all. GBDT is the best non-Deep Learning ML model. Nevertheless, GBDT is difficult to fit, prone to overfitting and the cost/benefit ratio compared to RF was not considered worthwhile (less than a 2% improvement). As far as DL models are concerned, ANN performed poorly and ANN_ens was only able to reach the RF baseline performance at a much higher training cost. Finally, the CNN part of CNN/RNN did not help to improve the performance and RNN_last: although it showed similar MAE to RNN_all, its standard deviation was higher.

Based on the results for SIV1 while using all the models considered, it was decided to expand more in-depth analysis only of RF and RNN_all while considering the other sector and variants of those models discussed in section III. In fact, Table 6 confirms that the ML approach works on other Nice sectors. Results on SIV2/3 completely corroborate preliminary results obtained on SIV1 with RF and RNN_all showing, respectively, 22% and 24% performance improvement compared to the Nice baseline.

Table 7 presents the results obtained when training the same models while preliminary applying the averaging windows pre-processing but without the new features engineering phase, i.e. without NFEP. Its effect was evaluated on the RF and RNN_all models (respectively RF without NFEP, RNN_all without NFEP) on both sectors. First, the Nice baseline with averaging windows slightly improves its global performance results compared to the results without averaging windows. More interestingly, comparing the ML models with averaging windows to the Nice baseline without averaging windows showed clear improvements: a 28% and 25% MAE decrease on SIV1, SIV2/3, respectively, for RF and 31.3% and 31.1% MAE decrease on SIV1, SIV2/3, respectively, for RNN_all.

Table 7 also shows the results encompassing the new features engineering phase (NFEP) in addition to the effect of averaging windows. It allows to obtain the best RF and RNN_all models for both SIVs, validating the global approach with the following results: 30.3% decrease on both SIV1 and SIV2/3, respectively, for RF and 31.3% and 32.6% MAE

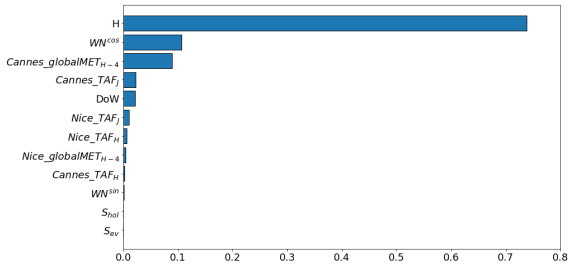
decrease on SIV1 and SIV2/3, respectively, for RNN_all compared to the Nice baseline without averaging windows. Compared to the former RF baseline in [1], our new state-of-the-art model RNN_all with averaging windows and NFEP allowed to decrease MAE by 12.2% and standard deviation by 10.9%. This demonstrates the added value of the averaging windows pre-processing to reduce GA entry count prediction error performed by the ML model when trained on these new data instead of rough entry counts.

In conclusion, the new features engineering phase allows us to take advantage of the new meteorological information. Compared to the Nice baseline, improvements are particularly notable for SIV2/3 global results, where the meteorological information is better correlated with the geography. The contribution of the new features engineering will be further analyzed next for the two best ML models: RF and RNN_all with NFEP and averaging windows. The analysis is performed on SIV1 and SIV2/3 data using two methods: a global analysis and a local analysis to provide explainability on each black box model. Note that from now on RNN_all model will be referred to as RNN.

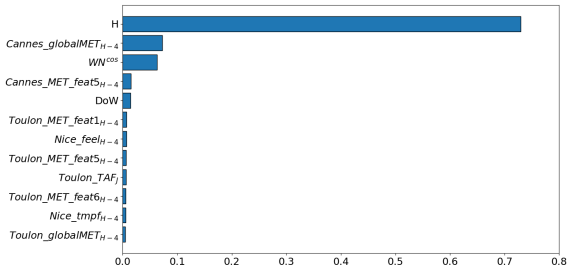
B. Global Analysis of the Best Models Using the Permutation Feature Importance Method

The global analysis presented here uses the first feature importance method based on Permutation Feature Importance (PFI) to evaluate the global impact of each feature on our model. This method measures the predictive value of a feature for any estimator by evaluating how the prediction error increases when a feature is removed from the input to the model. For each feature, this is implemented by shuffling randomly the values of the feature in the original dataset used for evaluation (D_{test}). The predictions with the original model on the "shuffled" dataset are compared through MAE metric indicating the importance of the feature.

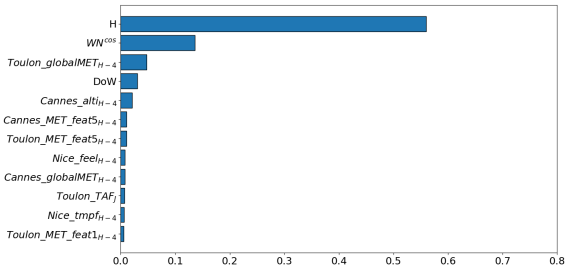
Results for the RF models are presented in Fig. 8. PFI for the former RF baseline on SIV1 [1] are compared with the new state-of-the-art RF model. H, WN^{cos}, DoW and $Cannes\_globalMET_{H\text{-}4}$ features display the same level of importance for both models. However, there were some significant differences: 8 newly integrated features ($Cannes\_MET\_feat5_{\text{H-4}}$, $Toulon\_MET\_feat1_{\text{H-4}}$, $Nice\_feel_{\text{H-4}}$, $Toulon\_MET\_feat5_{\text{H-4}}$, $Toulon\_TAF_{\text{J}}$, $Toulon\_MET\_feat6_{\text{H-4}}$, $Nice\_tmpf_{\text{H-4}}$, $Toulon\_globalMET_{H\text{-}4}$) are in the list of the 12 most important features for the new state-of-the-art SIV1 RF model with a global contribution of 6% (Figs. 8a and 8b). The PFI of RF for SIV2/3 is presented in Fig. 8c confirming that the new features' global contribution is more significant for these SIVs (12%).



(a) PFI for former RF baseline on SIV1.



(b) PFI for the best RF SIV1 model (12 most important features).

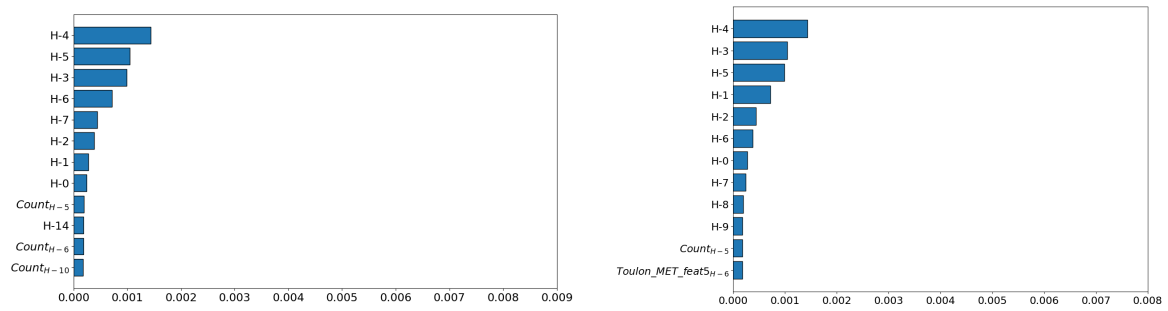


(c) PFI for the best RF SIV2/3 model (12 most important features).

Fig. 8 PFI comparison for the RF models.

Results for the RNN models are presented in Fig. 9 for both SIV1 and SIV2/3 models. These are not satisfactory as the method is not suitable to handle such a high number of features (7875 features taking into account all the past features over 1 week). Even if the features' importance values of the 12 most important features are very low, it can be

noted that H and $Count$ past and present features are the most important for both models. This limitation highlights the need for an alternative method to analyze the results, i.e., the use of the SHAP method proposed in the next subsection.



(a) PFI for the best SIV1 RNN. **(b)** PFI for the best SIV2/3 RNN (12 most important features).

Fig. 9 PFI comparison for the RNN models.

C. Global Analysis of the Best Models Using the SHAP Method

1. The SHAP Method

SHapley Additive exPlanations (SHAP) is a model-agnostic method well-recognized in the ML world as a global method to describe how features affect the prediction on average but also as a local method to explain individual predictions [25]. This method is based on Shapley values, a concept from Game Theory [26].

The Shapley value is calculated for each feature. This is the only method that satisfies the Efficiency, Symmetry, Dummy, and Additivity properties, which together can be considered as the definition of a fair payout attribution between the features. Applying this concept to a particular instance x predicted by our model, the Shapley value of the feature j, denoted by ϕ_j, indicates know how each feature contributed to the prediction of this instance compared to the average prediction for the dataset. The Shapley value of a specific feature value is the average change in the prediction that a coalition of considered features S gains when the specific feature is added to them. Then, the notion of the value function, denoted by $val_x(S)$, must be introduced. If S is a coalition of features (i.e., a subset of the features used in the model), then $val_x(S)$, called the worth of coalition S, describes the total expected sum of payoffs the members of S can obtain by cooperation. Its mathematical expression is the prediction for feature values in set S that are marginalized over features that are not included in set S and is given by

$$val_x(S) = \int f\left(x_1, ..., x_m\right) \mathrm{d}P_{x \notin S} - \mathbb{E}_X\left[f(X)\right] = \mathbb{E}_X\left[f(X) \middle| X = x_S\right] - \mathbb{E}_X\left[f(X)\right], \tag{9}$$

where f is the model to explain, m is the number of features, i.e., $x \in \mathbb{R}^m$. The Shapley value ϕ_j is obtained considering all possible different permutations in which the coalition S can be formed, then adding feature j to it, and considering that its contribution is the difference between $val_x(S \cup \{j\})$ and $val_x(S)$, i.e.,

$$\phi_j(x) = \sum_{S \subseteq \{1,...,m\} \setminus \{j\}} \frac{|S|!\left(m - |S| - 1\right)!}{m!}\left[val_x\left(S \cup \{j\}\right) - val_x\left(S\right)\right]. \tag{10}$$

Examining (10) reveals that the exact calculation of Shapley values can be intractable. SHAP first proposes an approach to efficiently approximate them. Specific and optimized algorithms have been proposed to tackle Tree based models (TreeSHAP) with a polynomial time algorithm that recursively keeps track of what proportion of all possible subsets flow down into each of the leaves of the tree and Neural Networks models (DeepSHAP) using the compositional nature property of the NN [25, 27]. Alternatively, SHAP suggests a local and a global method based on the Shapley value explanation that is represented as an additive feature attribution method. Specifically, the SHAP local method specifies an explanation model $\tilde{f}$ that approximates f as a linear model, with Shapley values features of the model, and

expressed as

$$f(x) \approx \tilde{f}(z') = \phi_0 + \sum_{j=1}^{M} \phi_j * z'_j \tag{11}$$

with M being the maximum coalition size (number of features or group of features), $z' \in \{0,1\}^M$ the coalition vector (1 means corresponding feature is in coalition and its value is related feature value of x, 0 means corresponding feature is not in coalition and related value is assigned randomly), and ϕ_j the Shapley value of feature j (or group of features j).

The latter method allows to define SHAP global importance of a feature defined as average of the Shapley values per feature across the whole dataset D, given by

$$I_j = \frac{1}{n} \sum_{i=1}^{n} \left| \phi_j^{(i)} \right|. \tag{12}$$

It is an alternative to permutation feature importance that relies on the magnitude of feature attributions instead of a decrease in model performance.

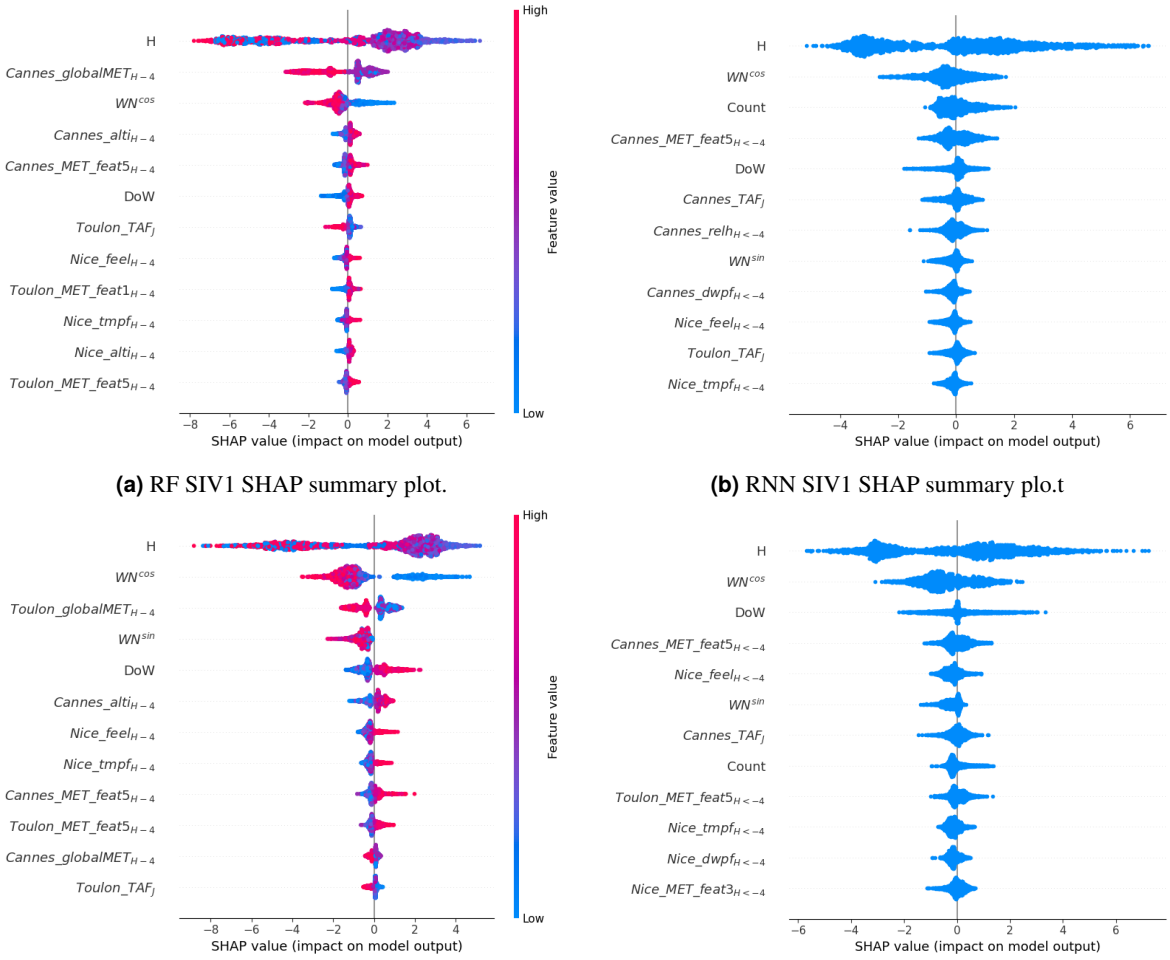
2. Evaluation of the SIV1 Models and Comparison with the Previous Baseline

In this section, the aim is to evaluate the RF and RNN models trained on SIV1, and to compare them with each other and with the former RF baseline [1].

Figures 10a and 10b depict an overview of the most important features for both the RNN and RF SIV1 models considering their SHAP importance on the test set. It confirms that, for both of them, H is the most important feature. However, apart from this feature, each model relies on very different information: $Count$ and meteorological past data (e.g. $Cannes\_MET\_feat5_{\text{H-4}}$, $Nice\_dwpf_{\text{H-4}}$, $Nice\_relh_{\text{H-4}}$, $Nice\_tmpf_{\text{H-4}}$) have a major contribution for the RNN whereas the RF can only rely on present data. It shows that the meteorology evolution over time is crucial for the GA traffic evolution in the medium term perspective. $Count$ features help to smooth the predictions over time, probably explaining the major contribution to the standard deviation reduction of RNN presented in section IV.A.

Figure 11 presents another view of the features contribution based on their SHAP importance on the test set. In this plot, features are grouped by category. The advantage of the SHAP method is its additivity property that allows to group each feature contribution and to calculate the global contribution of the group of features, which was not possible with the PFI method. Figures 11a and 11b consider a first breakdown of all the features: the first group is related to the time slot (H), the second to the weather data available 4 hours before the slot to predict ($Weather\_data_{H-4}$), the third deals with the week number information (WN), the fourth with weather data available one day before the slot to predict ($Weather\_data_{J-1}$), the fifth considering all the information about the day (DoW_and_event), and the last group (specific to the RNN model) considers all the past counts information ($Count$). These figures confirm the importance of the weather information 4 hours before the slot to predict, reaching more than 33% SHAP importance for the RNN model. It confirms that GA counts prediction is a difficult task since it is heavily affected by meteorological changes, which can be so uncertain that pilots wait until the very last moment to decide whether to fly or not.

Figures 11c and 11d present the same breakdown splitting the test set into quartiles with respect to the target entry counts prediction of each slot. It shows that the importance of H group decreases when count predictions are increasing which seems strongly linked to the fact that very low entry counts mainly occur early in the morning or in the evening. Moreover, $Weather\_data_{H-4}$ group importance exceeds H group importance for Q4 quartile. WN and $Weather\_data_{J-1}$ groups have more importance for Q2 and Q3 quartiles. Figures 11e and 11f consider a second breakdown of the features: the new ones identified in the present article and the previous features made available in [1]. The new features contribution is significant for RF with around 20% SHAP importance and major for RNN with 45% SHAP importance.



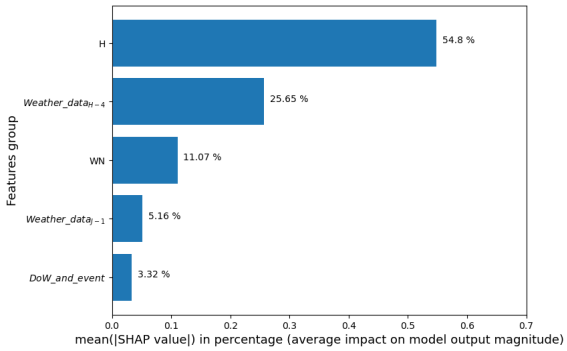
(a) RF SIV1 SHAP summary plot.

(b) RNN SIV1 SHAP summary plo.t

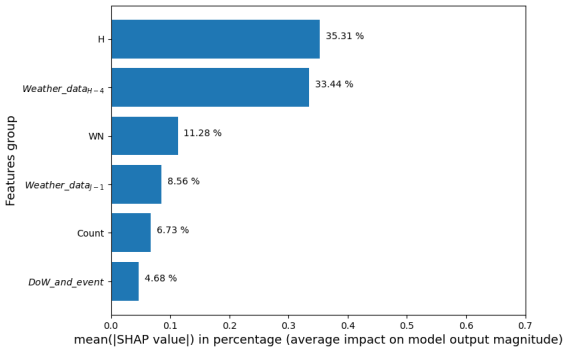
(c) RF SIV2/3 SHAP summary plot.

(d) RNN SIV2/3 SHAP summary plot.

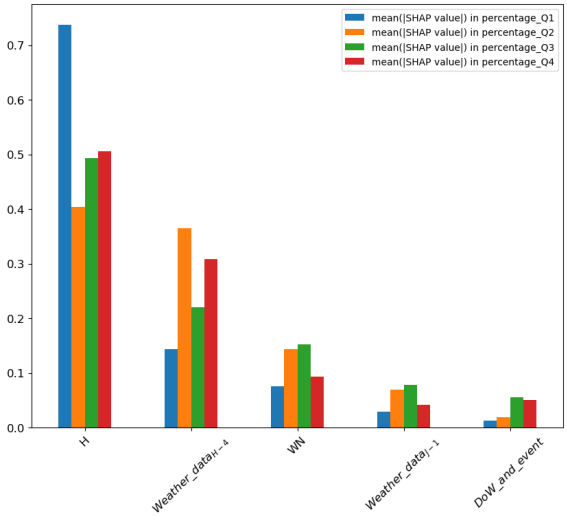
Fig. 10 SHAP summary plot (12 most important features) for the SIV1 and SIV2/3 models features.



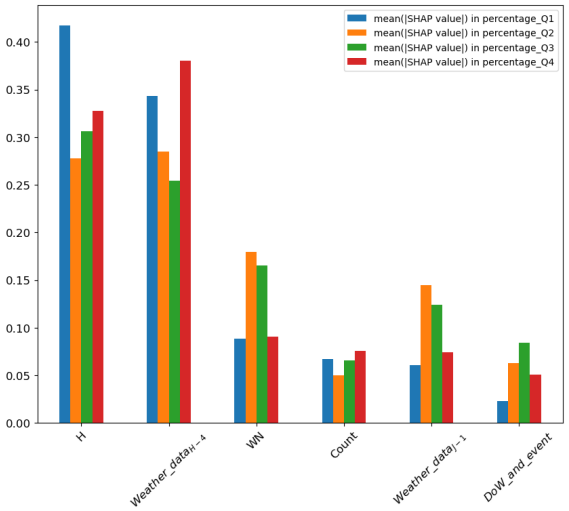
(a) RF SIV1 SHAP summary plot - features type category.



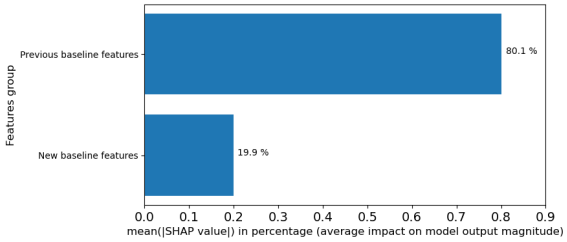
(b) RNN SIV1 SHAP summary plot - features type categ.



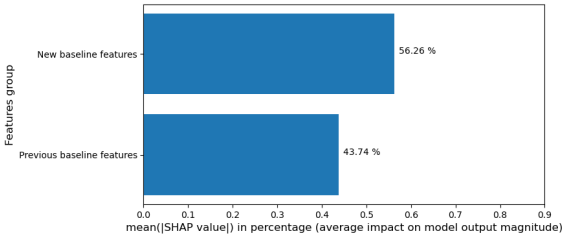
(c) RF SIV1 SHAP summary plot - feat. type categ. per quartile.



(d) RNN SIV1 SHAP summ plot - feat type categ per quart.



(e) RF SIV1 SHAP summary plot - new/former category.



(f) RNN SIV1 SHAP summary plot - new/former categ.

Fig. 11 SHAP summary plot for the SIV1 models features grouped by category.

3. Evaluation of the SIV2/3 Models

This part is related to the SIV2/3 RF and RNN models evaluation with the SHAP method. Figures 10c and 10d present the most important features for both models. It is not surprising to see that H remains the most important feature but the rest of the classification highlights many differences, with the SIV2/3 RF relying more on Toulon weather than on Cannes and Nice. WN features also seem to be more significant for this model which shows that the seasonality has much more impact for these SIVs. This is not surprising considering their geographical location. The SIV2/3 RNN presents a pattern very close to the SIV2/3 RF's with the following differences: DoW feature has more impact as the third most important feature of the model. Moreover, even if $Count$ feature is present in the RNN most important features classification, it has clearly less importance than for the SIV1 RNN model. It can be finally noted that the Bastia meteorological information does not seem to have any importance for any of the models or SIVs. This is surprising for SIV2/3 models if we consider their geographical location.

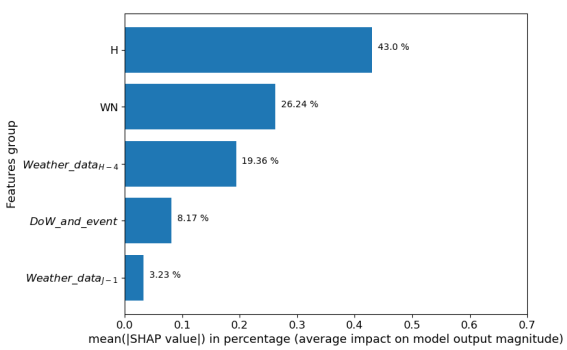
Figure 12 presents the features contribution based on their SHAP importance grouped by category as in Fig. 11. Figures 12a and 12b consider the same features breakdown proposed in Figs. 11a and 11b. As for SIV1, it shows the importance of the weather information 4 hours before the slot to predict, reaching 32% SHAP importance for the RNN model ($Weather\_data_{H-4}$). However, the SIV2/3 RF model less relies on weather information ($Weather\_data_{H-4}$, $Weather\_data_{J-1}$ groups) as much as the SIV1 RF model. WN group is much more important for these models: this group reaches 11% SHAP importance for both SIV1 models and reached 15% and 26% for the SIV2/3 RNN and RF models respectively. Finally, the SIV2/3 RNN model relies more on DoW_and_event group and less on $Count$ than the SIV1 RNN model as already seen before. Figures 12c and 12d present the same breakdown splitting the test set into quartiles with respect to the target entry counts of each slot. It confirms the observation made on SIV1 as the distribution over the quartiles is very similar for both SIVs: the importance of H group decreases as counts increase, and the WN group is more important for Q2 and Q3 quartiles.

D. Local Analysis of the Best Models

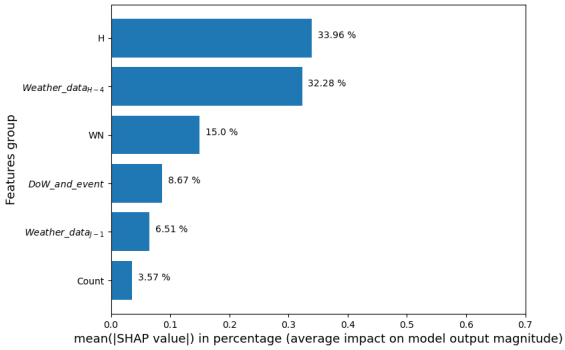
This section compares all the models with each other and with the Nice baseline on specific examples. The purpose is to define an approach to better understand model prediction to give ATCo more confidence in them as, to date, one remaining limitation to using ML-based tools in operation is their lack of transparency.

First, a focus is made on a specific day, the 15th of May 2022, to compare the results of all the reference models. Figures 13a and 13b compare each RNN and RF performance by displaying the absolute error between the model prediction and the true entry count value for each slot of the day on SIV1 and SIV2/3 respectively. For SIV1, the former RF and the Nice baseline performance are also displayed, showing the improvements of the new RNN and RF models. The performance of the RNN is also highlighted by the error over the day slots variance for both SIVs (orange bars). These figures are complemented with Figs. 13c and 13d that present the RNN and RF performance on the full month of May 2022: the cumulative absolute error over each day is presented for each model and each SIV. These figures illustrate the improvements of both the RF and RNN models over the month compared to the former reference baselines: the former RF for SIV1 and the Nice baseline for SIV2/3.

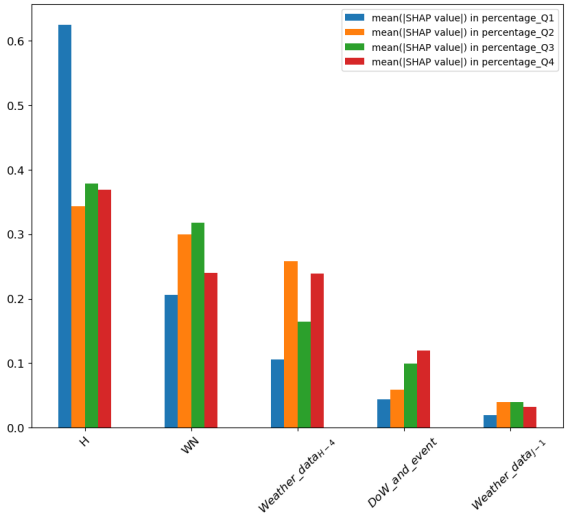
Finally, waterfall plots of the Shapley values calculated over the test set are proposed to support operational decisions and complement the model predictions. For a specific slot, it starts with the expected value of the predictions over the set and gives information on the features impacting the prediction of the specific slot the most. For instance, Figs. 14a and 14c show how the RF and RNN models respectively have taken into account main features to come to their final prediction for the 12-hour slot on 15th May 2022, 12h slot (entry counts: 10.754 for RF / 12.795 for RNN) starting from the expected value of all the predictions (5.839 for RF / 5.61 for RNN). For both of them, H has a major impact on the target prediction increase (+2.2 for RF / +2.36 for RNN). Meanwhile, Cannes TAF (Cannes_TAF$_J$ for RF and Cannes_TAF$_H$ for RNN) has a slight impact on the target prediction decrease (-0.25 for RF for / -0.29 for RNN). The authors are convinced that this kind of tool can bring real added value to operational staff as it helps understand the black box model decision, gives confidence in the prediction, and helps the operator change the decision if he considers that the model has missed some crucial information not presented in the waterfall plot. It can also help identify easily if the model is performing poorly in specific situations and can bring further ideas for potential complementary model training to take these situations into account.



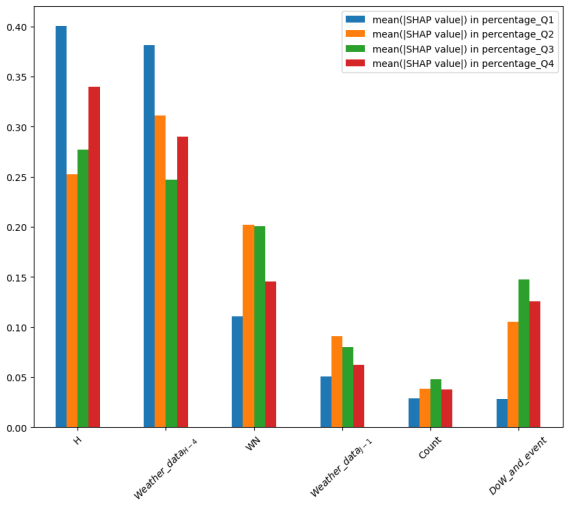
(a) RF SIV2/3 SHAP summ plot - features type category.



(b) RNN SIV2/3 SHAP summ plot - features type categ.

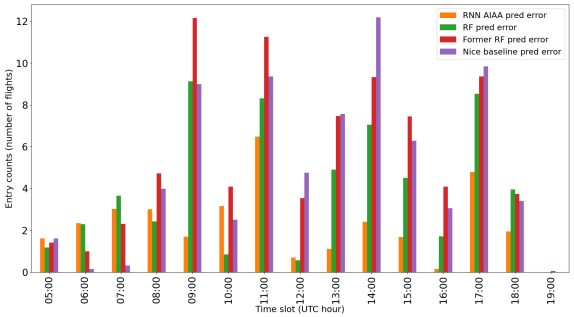


(c) RF SIV2/3 SHAP summ plot - feat. type categ. per quartile.

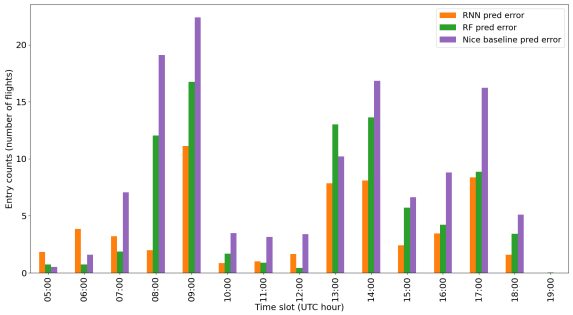


(d) RNN SIV2/3 SHAP summ plot - feat type categ per quart.

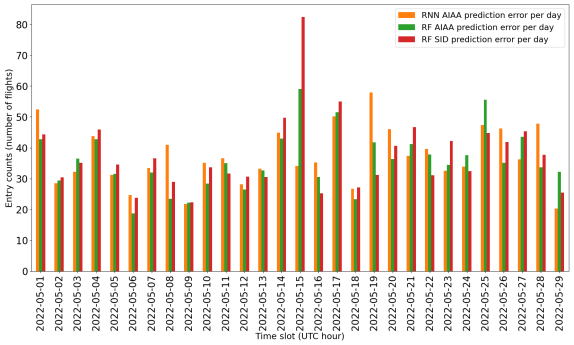
Fig. 12 SHAP summary plot for the SIV2/3 models features grouped by category.



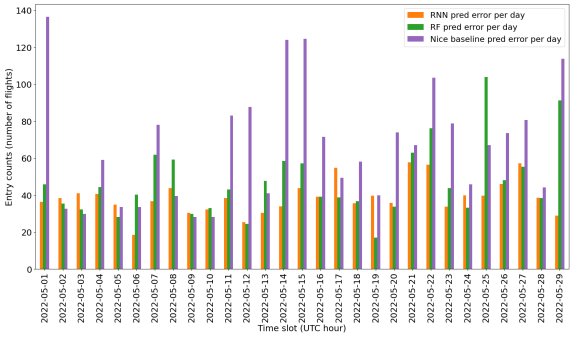
(a) SIV1 results for Thursday 15 May 2022.



(b) SIV2/3 results for Thursday 15 May 2022.

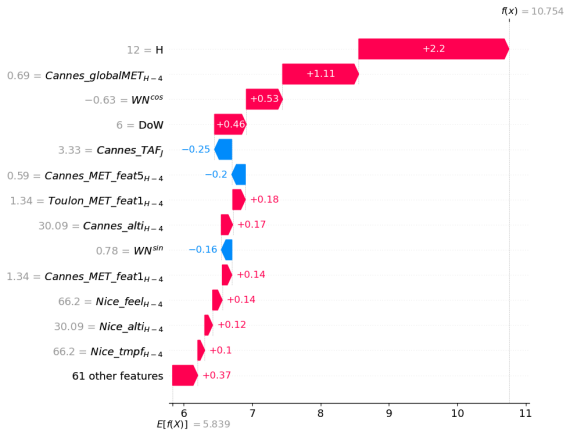


(c) SIV1 results for May 2022.

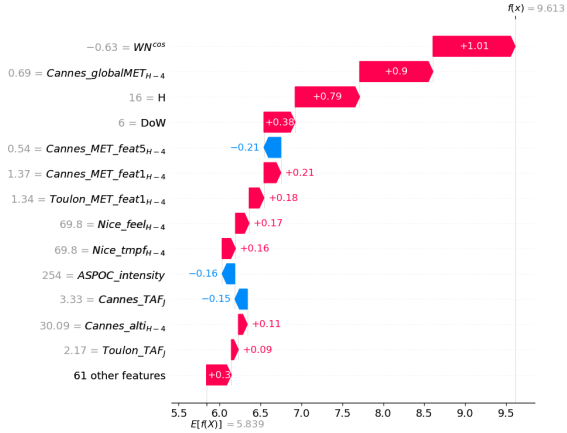


(d) SIV2/3 results for May 2022.

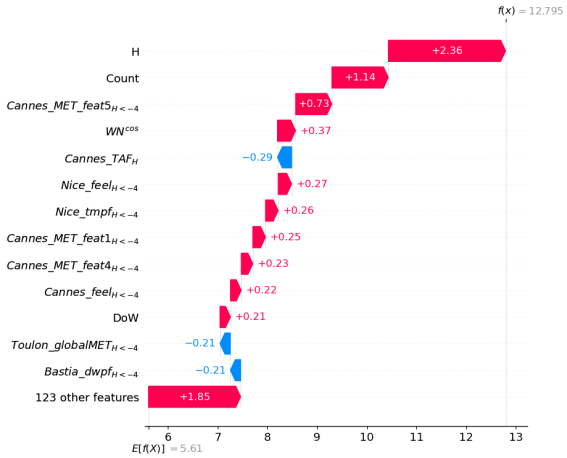
Fig. 13 Illustrative example of results comparison.



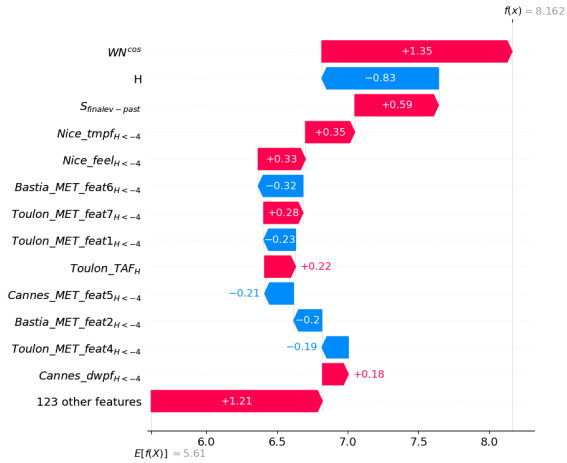
(a) RF waterfall plot, 15 May 2022, 12 h.



(b) RF waterfall plot, 15 May 2022, 16 h.



(c) RNN waterfall plot, 15 May 2022, 12 h.



(d) RNN waterfall plot, 15 May 2022, 16 h.

Fig. 14 Example of waterfall plots for the SIV1 models.

V. Conclusions

The objective of this paper was to propose an updated and state-of-the-art methodology to predict GA entry counts of flights in advance, expanding the results in [1]. It has been shown that new models based on Recurrent Neural Networks with the support of a new features engineering phase based on Deep Learning and the averaging windows pre-processing improve the performance of the proposed Machine Learning approach as tested on the Nice TCC sectors SIV1 and SIV2/3. This resulted in improvements such as 31.3% and 32.6% MAE decrease on SIV1 and SIV2/3, respectively, compared to the Nice baseline currently in operation. Compared to the RF presented in [1], and using the same set of historical data, the RNN_all model with averaging windows and NFEP allowed to decrease of MAE by 12.2% and standard deviation of the prediction error by 10.9%. In addition, this study addressed the explainability of the ML models using the SHAP framework based on Shapley values theory. It provides the operational staff additional information to better understand the ML model predictions which facilitate its operational use as a support to ATCo decisions.

In future work, it could be valuable to consider the following areas: new data exploration (e.g. wider dataset with additional years, detailed COVID-19 period analysis), assessment of new features (e.g. flying schools booking slots, flights trajectory), an additional framework based on a level of traffic prediction (classification task) instead of a regression task. This raises the question of the ML cost function that probably can be adapted in the future with a second training criteria based on the minimization of the entry counts error on the complete day keeping the distribution of the error over the day slots variance as minimal. A further step towards operational use would be to address the quantification of the uncertainty of the ML prediction with Bayesian Neural Networks [28] and on the covariate shift detection [29].

Acknowledgments

This work was carried out in co-operation with DSNA, METEO FRANCE and Metsafe. The authors thank Nice TCC staff for providing operational expertise and data.

References

[1] Abecassis, A., Delahaye, D., and Idan, M., "A Machine Learning Framework to Predict General Aviation Traffic Counts - A Case Study for Nice Cote D'Azur Terminal Control Center," *12th SESAR Innovation Days*, 2022, pp. 1–9.

[2] Chakraborty, D., and Elzarka, H., "Advanced machine learning techniques for building performance simulation: a comparative analysis," *Journal of Building Performance Simulation*, Vol. 12, 2018, pp. 1–15. https://doi.org/10.1080/19401493.2018.1498538.

[3] Goodfellow, I. J., Bengio, Y., and Courville, A., *Deep Learning*, MIT Press, Cambridge, MA, USA, 2016.

[4] IOWA State University, "Iowa Environmental Mesonet," https://mesonet.agron.iastate.edu/, Accessed: September 2023.

[5] Ballester, G., "OGIMET," https://www.ogimet.com, Accessed: September 2023.

[6] Mikolov, T., Chen, K., Corrado, G., and Dean, J., "Efficient Estimation of Word Representations in Vector Space," *Proceedings of Workshop at ICLR*, Vol. 2013, 2013.

[7] Bank, D., Koenigstein, N., and Giryes, R., "Autoencoders," *arXiv*, 2021. http://arxiv.org/abs/2003.05991.

[8] Yu, W., Kim, I. Y., and Mechefske, C., "Analysis of different RNN autoencoder variants for time series classification and machine prognostics," *Mechanical Systems and Signal Processing*, Vol. 149, 2021, p. 107322. https://doi.org/https://doi.org/10.1016/j.ymssp.2020.107322.

[9] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimization," *arXiv*, 2017. http://arxiv.org/abs/1412.6980.

[10] Meteo France, "Meteo France," https://https://meteofrance.com//, Accessed: September 2023.

[11] Rebaï, K., and Warnier, M., "Metsafe," https://userportal.metsafecloud.com/, Accessed: September 2023.

[12] Géron, A., *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly Media, 2019. https://books.google.fr/books?id=HHetDwAAQBAJ.

[13] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Curran Associates, Inc., 2017, pp. 1–9. https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[14] Ganaie, M., Hu, M., Malik, A., Tanveer, M., and Suganthan, P., "Ensemble deep learning: A review," *Engineering Applications of Artificial Intelligence*, Vol. 115, 2022, p. 105151. https://doi.org/10.1016/j.engappai.2022.105151.

[15] Hochreiter, S., and Schmidhuber, J., "Long Short-term Memory," *Neural computation*, Vol. 9, 1997, pp. 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

[16] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y., "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv*, 2014. http://arxiv.org/abs/1406.1078.

[17] Zhao, L., Cheng, B., and Chen, J., "A Hybrid Time Series Model based on Dilated Conv1D and LSTM with Applications to PM2.5 Forecasting," *Aust. J. Intell. Inf. Process. Syst.*, Vol. 17, 2019, pp. 49–60.

[18] Ioffe, S., and Szegedy, C., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," , 2015. http://arxiv.org/abs/1502.03167.

[19] Ba, J. L., Kiros, J. R., and Hinton, G. E., "Layer Normalization," *arXiv*, 2016. http://arxiv.org/abs/1607.06450.

[20] Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B., "Algorithms for Hyper-Parameter Optimization," *Proceedings of the 24th International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, 2011, p. 2546–2554.

[21] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M., "Optuna: A Next-Generation Hyperparameter Optimization Framework," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Association for Computing Machinery, New York, NY, USA, 2019, p. 2623–2631. https://doi.org/10.1145/3292500.3330701, URL https://doi.org/10.1145/3292500.3330701.

[22] Breiman, L., "Bagging Predictors," *Machine Learning*, Vol. 24, No. 2, 1996, pp. 123–140.

[23] Chen, T., and Guestrin, C., "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Association for Computing Machinery, New York, NY, USA, 2016, p. 785–794. https://doi.org/10.1145/2939672.2939785, URL https://doi.org/10.1145/2939672.2939785.

[24] Salehin, I., and Kang, D.-K., "A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain," *Electronics*, Vol. 12, No. 14, 2023. https://doi.org/10.3390/electronics12143106.

[25] Lundberg, S. M., and Lee, S., "A unified approach to interpreting model predictions," *arXiv*, 2017. http://arxiv.org/abs/1705.07874.

[26] Shapley, L. S., *A Value for N-Person Games*, RAND Corporation, Santa Monica, CA, 1953. https://doi.org/10.7249/P0295.

[27] Lundberg, S. M., Erion, G. G., and Lee, S.-I., "Consistent Individualized Feature Attribution for Tree Ensembles," *arXiv*, 2019. http://arxiv.org/abs/1802.03888.

[28] Jospin, L. V., Buntine, W. L., Boussa, F., Laga, H., and Bennamoun, M., "Hands-on Bayesian Neural Networks - a Tutorial for Deep Learning Users," *arXiv*, 2020. http://arxiv.org/abs/2007.06823.

[29] Alberge, F., Feutry, C., Duhamel, P., and Piantanida, P., "Detecting Covariate Shift with Black Box Predictors," *2019 26th International Conference on Telecommunications (ICT)*, 2019, pp. 324–329. https://doi.org/10.1109/ICT.2019.8798827.