Journal of Open Aviation Science (2024), Vol.**2**
doi:10.59490/joas.2024.7890

JOAS

PROCEEDINGS | *The 12th OpenSky Symposium*

Data-Driven Prediction of Aircraft Holding Times Using OpenSky Data

Michele Vella [*,1], Jason Gauci [1] and Alexiei Dingli [1]

University of Malta, Msida, Malta

*Corresponding author: michele.vella.15@um.edu.mt

(Received: 30 Oct 2024; Revised: 12 Feb 2025, 3 Jun 2025; Accepted: 24 Jun 2025; Published: 10 Jul 2025)
(Editor: Manuel Waltert; Reviewers: Ramon Dalmau, Timothé Krauth, Max Li)

Abstract

As global air traffic increases, major hubs such as London Heathrow and Gatwick experience increasing congestion, leading to the frequent queuing of aircraft in holding stacks. This study employs machine learning (ML) techniques to predict aircraft holding times at London Heathrow, aiming to enhance flight management and reduce congestion in the Terminal Maneuvering Area (TMA) by enabling adjustments to flight trajectories and speeds earlier in the flight, rather than upon arrival. Leveraging historical data - including surveillance data from the OpenSky Network - from April 2023 to March 2024, the study adopts the LightGBM and LSTM ML frameworks to develop two sets of predictive ML models: one set of regression models to predict the holding time of individual flights up to 60 minutes from the London TMA; and one set of time series regression models to predict the average holding time in different holding stacks. Test results of the regression models show that the models trained with LightGBM have the best performance, with minimum RMSE and MAE values of 2.25 and 1.50 minutes, respectively. On the other hand, the results of the time series regression models show better performance by the models trained with LSTM, with average RMSE and MAE values of 2.41 and 1.47 minutes, respectively. In conclusion, this research highlights the effectiveness of ML in predicting aircraft airborne holding times, offering significant benefits to pilots, air traffic controllers, and flight planners. The successful application of these models could lead to substantial improvements in flight efficiency and reduced environmental impact from fewer delays and less fuel consumption.

Keywords: OpenSky; Heathrow; Holding Patterns; LightGBM; LSTM

Abbreviations: BIG: Biggin holding stack, BNN: Bovingdon holding stack, CDO: Continuous Descent Operation, LAM: Lambourne holding stack, LightGBM: Light Gradient-Boosting Machine, LSTM: Long Short-Term Memory, MAE: Mean Absolute Error, ML: Machine Learning, OCK: Ockham holding stack, OSN: Open Sky Network, PMS: Point Merge System, RMSE: Root Mean Square Error, SHAP: SHapley Additive exPlanations, TMA: Terminal Manoeuvring Area

1. Introduction

Air travel is becoming more popular, and, as a result, a heavy increase in air traffic is foreseen and can be felt worldwide. In 2024, air traffic increased between 10% and 20% when compared to 2019 in large parts of European airspace, the highest growth ever in terms of air traffic in Europe. In June 2024, there were on average 33,671 daily flights compared to 32,010 in June 2023 - a 5.2% increase across the network as a whole [1].

One of the consequences of this development is that the airspace is becoming more congested, particularly in Terminal Airspace, with aircraft occasionally having to wait in a holding stack - from

a few minutes to over an hour - before being cleared to land. This is particularly the case of busy hub airports such as London Heathrow and London Gatwick. Apart from holding, methods such as the Point Merge System (PMS) and tromboning can also be used to manage traffic in a Terminal Airspace.

Point Merge was designed by the EUROCONTROL Experimental Centre to handle high traffic loads without the need for radar vectoring. It is based on a specific P-RNAV route structure, consisting of a point (the merge point) and pre-defined legs (the sequencing legs). Point Merge is now operational at 56 airports in 23 countries and 4 continents, such as Istanbul, Shanghai, and Tokyo. [2].

Holding occurs when the rate of aircraft arriving in the Terminal Manoeuvring Area (TMA) exceeds the runway capacity i.e. there is a demand-capacity imbalance. In this case, aircraft are asked to enter a holding stack - a 'racetrack' pattern over a fixed point - until they are cleared to land. Runway capacity depends on many factors such as runway configuration; arrival/departure mix; weather conditions at the airport; and noise abatement procedures. This creates traffic congestion and delays. Additionally, holding occurs at relatively low altitudes (where the air is denser), resulting in higher fuel emissions and noise pollution. Noise pollution has a bigger impact when holding stacks are situated in the vicinity of residential areas. With today's challenge of global warming, it is imperative to decrease these emissions by using innovative technological solutions.

Today, pilots have very limited information to determine, in advance, how likely they are to enter a holding stack at the destination and, if so, how long they are likely to hold for. Having this information would be beneficial to pilots as it would help them with their fuel planning, and it would give them a better idea of the time delay they are likely to encounter in the TMA.

Air Traffic Controllers (ATCOs) would also benefit by having advance knowledge of the holding times expected by individual aircraft, or the average holding time in a holding stack. Such information would help them to plan their resources more effectively. Furthermore, ATCOs could take action to absorb any delays earlier in a flight - such as by changing an aircraft's trajectory or speed enroute - instead of requiring the flight to hold at the destination. This would also facilitate Continuous Descent Operations (CDO) which are less fuel-intensive than conventional step-down descents.

This paper investigates the use of supervised Machine Learning (ML) techniques to forecast aircraft holding times - both for individual flights and particular holding stacks - using London Heathrow Airport (IATA: LHR/ICAO: EGLL), and the London TMA, as a case study. London Heathrow was selected because it only has two runways, which makes it very challenging to manage the traffic load. As a result, about 36% of all arrivals spend time in a holding stack before landing [3]. London Heathrow has four holding stacks - called Bovingdon (BNN), Lambourne (LAM), Ockham (OCK), and Biggin (BIG) - as can be seen in Figure 1 . BNN, OCK, and BIG maintain a right-hand direction during the holding turn, whereas LAM follows a left-hand direction.

The rest of the paper is organised as follows. Section 2 gives a brief overview of the work that has been done in relation to this study and Section 3 describes the methodology used. Section 4 presents and analyses the results obtained. Finally, key conclusions and areas of future work are addressed in Section 5.

2. Related Works

Various studies have been conducted on the prediction of aircraft arrival times at their destination, but very few studies consider TMA delays due to holding flights, or explicitly try to predict or model holding time. In some cases, holding flights are even considered as outliers and are removed [4]; however, this reduces dataset representativeness.

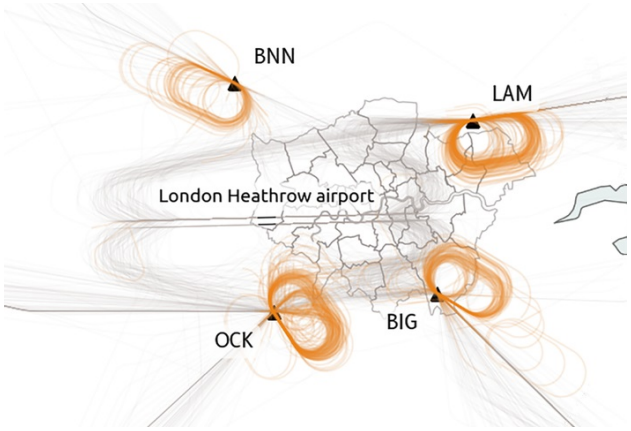

Figure 1. London Heathrow holding stacks [3].

ML algorithms to predict aircraft Estimated Time of Arrival are proposed in [5]. Here, the authors state that flights with holding patterns are not considered as outliers and are taken into account by calculating total flight distances from the trajectories and using them to train the models. The study uses random forests and deep Neural Networks for model training. The models can predict Estimated Time of Arrival with an MAE of less than 6 minutes and 3 minutes, immediately after departure and immediately after TMA entrance, respectively. One strength of these models is that they can be applied to other airports (with similar runway configurations) without modifications. A limitation is that departure traffic, as well as airport and airspace congestions, are not considered.

In [6] a study was carried out to predict holdings and delays in the TMA as part of an investigation into the implementation of an Extended Arrival Manager at Singapore Changi Airport. The CatBoost ML algorithm was adopted to predict holding times at distances of 200 NM from the airport. The inputs to the model are flight information, flight parameters, flight paths, weather, surrounding traffic, and airport performance metrics. In [7], the authors improve on the work in [6] to predict holding time and TMA delay at distances of up to 500 NM from the airport, achieving average RMSE and MAE values (for holding time prediction at 500 NM) of 0.978 minutes and 0.396 minutes respectively. The authors also propose speed control strategies to absorb TMA delays in the cruise phase. One limitation of [7] is that it was only tested at Singapore Changi Airport, so there is no guarantee that it would perform well at other airports.

In [8] the authors use a deep Convolution Neural Network based module to automatically detect holding-related features from trajectory images and other inputs. One of the advantages of this approach is that the trajectory images capture different types of information about an aircraft - including its position, speed, and relative position to other aircraft - reducing the need for complex holding-related feature engineering. The output of the Convolution Neural Network module is then used to predict the aircraft landing time for aircraft entering the TMA.

A recent research article [9] proposes a data-driven method to label airborne holdings based on causal factors. This research found that about one-fourth of 30-minute time intervals with airborne holdings are due to weather-related factors, with decreased visibility, strong winds and convective weather being the main contributors [9]. In particular, these weather-related issues account for roughly 40% of the total fuel consumption associated with these operations. This highlights how weather influences holding usage and its impact on fuel consumption and emissions.

In [10] an innovative linear programming approach is presented to schedule ground delay and air-

borne holding to prevent airspace capacity violations. The simulation results indicates a 13% decrease in fuel expenses compared to a rough estimate of existing practices as of 2009. A comparison between the suggested as-needed replanning technique and a comparable method employing fixed frequency replanning reveals a usual cost reduction of 1% to 2%, with potential reductions reaching up to 20% in certain scenarios.

As mentioned earlier in this section, there is a lack of studies - including ones that exploit ML - to predict airborne holding times. To address this gap, this study explores the application of ML techniques - specifically LSTMs and LightGBM - to this problem. As far as the authors are aware, these techniques have not been applied to this problem in prior work.

3. Methodology

This section first looks at how the historic data was acquired and preprocessed to generate training and test datasets that are suitable for the proposed application. Then, it describes the supervised ML models that were developed and trained to predict airborne holding time, including the ML techniques used, and associated hyperparameters.

3.1 Data Acquisition

For this study, three main types of historical data were acquired: traffic surveillance data, airport weather data, and airport delay data.

Raw Automatic Dependent Surveillance-Broadcast data was acquired from the OpenSky database [11] using the Traffic API [12]. This data includes parameters such as: ICAO24 identifier, aircraft latitude and longitude, altitude, airspeed, track, and time. The data was limited to data points within a bounding box with a size of 1380 NM by 1320 NM and centred on EGLL. The bounding box was defined in such a way that all the data points in the box correspond to flights that are within 1 hour from the London TMA boundary. Furthermore, only data corresponding to flights destined for EGLL were considered. Data was downloaded for a period of twelve months from April 2023 till March 2024, corresponding to a total of 222,975 arrivals at Heathrow. To improve download efficiency, data was downloaded at 30-second intervals. Each flight was uniquely identified using the Traffic API methods. Figure 2 shows a heat map of the raw aircraft positions acquired for arrival flights at Heathrow.

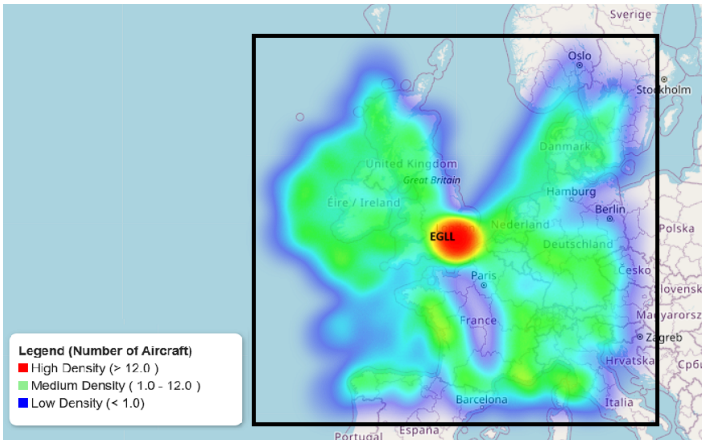

Figure 2. Heatmap of aircraft arriving at Heathrow on the 16th of March 2024 (The bounding box is also shown)

Weather has a great impact on aircraft holding times. For example, wind direction affects landing configuration and aircraft separation due to wake turbulence. Thunderstorm activity can result in runway closure and force aircraft to enter a hold. Low visibility reduces the Runway Visual Range (RVR) and triggers Low Visibility Procedures (LVO) which increase aircraft separation and cause disruptions. Every 30 minutes, an airport generates a Meteorological Aerodrome Report (METAR), which consists of actual weather data including: wind speed/direction; visibility; temperature and dew point; cloud density; and any weather phenomena in the vicinity of the airport. Historical METAR data for EGLL - for the period from April 2023 to March 2024 - was obtained from the Iowa Environmental Mesonet [13].

Another factor that can have an impact on holding times is airport delay. Airport delay information for Heathrow - for the period considered in this study - was obtained using the AeroDataBox[1] application of Rapid API[14] and is in the form of a delay index that is calculated separately for departing and arriving aircraft. The index is calculated every 15 minutes based on flight movements at the airport in the preceding two hours. The index can take values between 0 and 5; the higher the value, the more severe the delay experienced by the airport. As a guideline, the delay index is at its maximum value when 50% of the batches of flights are delayed for 90+ minutes (or cancelled). The AeroDataBox application was also used to obtain the number of cancelled arrival flights in the preceding two hours.

3.2 Detection of Holding Flights

Figure 3 shows a snapshot of the trajectories of flights in the dataset downloaded from OpenSky. The four holding stacks of EGLL are clearly visible in this plot. First, the data was filtered for each holding stack by applying a bounding box around each holding stack. For each holding stack, the bounding box was defined manually in such a way that it captures the extremities of the holding stack as defined in the enroute charts of the UK NATS AIP [15]. Subsequently, an algorithm was developed to automatically detect each flight in the dataset that entered a holding pattern, and to measure the duration of its holding. This involves examining changes in the heading of an aircraft over time to precisely pinpoint when it enters and exits a holding pattern. The algorithm is described in the rest of this section.

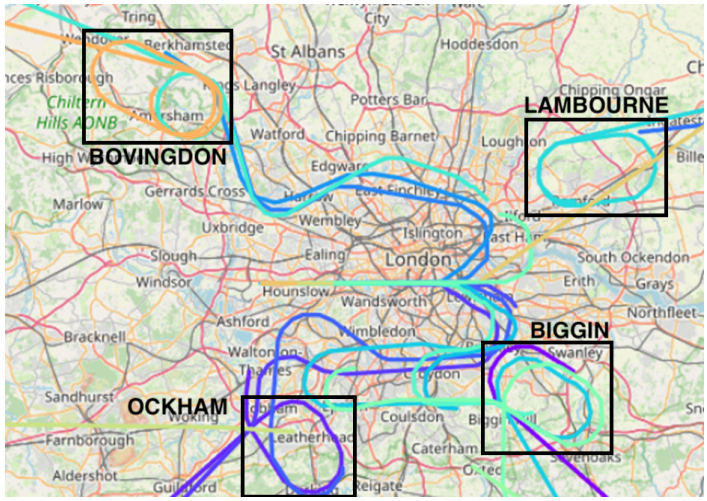

Figure 3. Trajectories of arrivals at EGLL. (The solid black lines represent bounding boxes around the holding stacks)

[1] A paid subscription is needed to download this data.

To detect a holding flight, the heading of a flight at each time step is compared to its heading in the previous time step (i.e. 30 seconds prior) to identify any difference. If a discrepancy of 20 degrees (or more) is detected, it is assumed that the aircraft has entered a holding pattern, and a timer starts. Subsequently, if a deviation of 5 degrees (or more) is detected in the direction opposite to that of the holding pattern, or if the aircraft is no longer in the bounding box corresponding to the holding stack, it is assumed that the aircraft has left the holding stack, and the timer stops. The holding time of the aircraft is then considered to be equal to the value of this timer. This method of detecting a holding flight is shown graphically in Figure 4.

This process was repeated for each arrival flight and, if the flight did not encounter a hold detection, its holding time was set to 0. The algorithm is designed to take into account the direction of the hold manoeuvre. The direction of holding at each stack can be observed in Figure 1.

It should be noted that, due to the sampling interval used, the error in the calculated holding time can be up to 1 minute. Unfortunately, no ground truth data was available to assess the accuracy of the proposed algorithm. To validate it, a small number of holding flights were selected and their holding time - as calculated by the algorithm - was compared with the holding time as measured manually by visualising each flight's trajectory. The algorithm was validated further by randomly selecting 100 flights and comparing the holding time calculated by the proposed holding detection algorithm with the holding time found using the corresponding method in the Traffic API [16]. The mean and standard deviation of the difference in holding time between the two methods is 1.2 minutes and 2.3 minutes, respectively. It was also observed that 12 flights were correctly detected (as holding flights) by the proposed algorithm but not by the Traffic API.

Figure 5 shows a histogram of the monthly number of flights in each holding stack over the entire dataset. One can note that the Bovingdon (BNN) stack was the busiest from April until June and September, while the Lambourne (LAM) stack was the busiest holding stack for the remaining months.

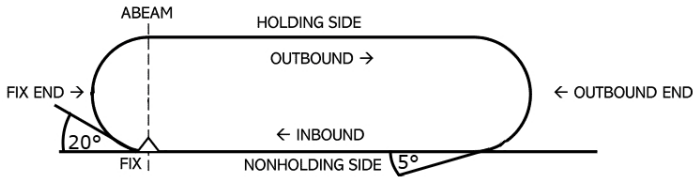

Figure 4. Holding Detection

3.3 Data Manipulation and Preprocessing

The data described in Sections 3.1 and 3.2 was manipulated to derive additional parameters and to transform it into a format that can be processed by the ML algorithms described in Section 3.4. The various manipulations are described in the rest of this section.

Two datasets were created as shown in Tables 1 and 2. In the first dataset, each entry corresponds to an individual flight, with the target parameter (aka 'output' aka 'dependent variable') being the flight's own holding time. In the second dataset, each entry corresponds to a specific 15-minute time interval, with the target parameter being the average holding time for each of the four holding stacks in that time interval. The second dataset is essentially a time series dataset. Dataset 1 consists of a total of 222,975 flights, with each flight characterized by 53 distinct features. Dataset 2 comprises 34,848 intervals, each lasting 15 minutes, with each interval characterized by 56 distinct features.

Non-numeric variables can only be handled by ML algorithms following some kind of transforma-

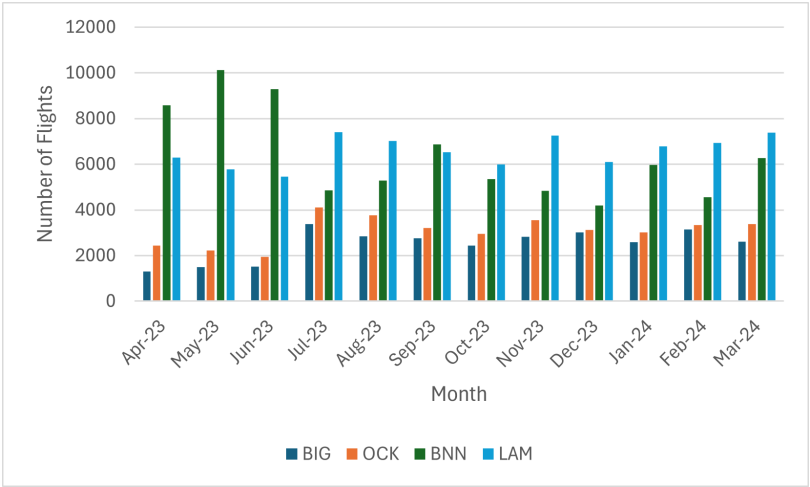

Figure 5. Number of flights in each holding stack

tion; for this reason, parameters with string values had to be converted to a numerical score or encoded (as can be observed in Tables 1 and 2). For the LSTM models, one hot encoding was applied to several parameters whereas, in the case of LightGBM, the Fisher [17] integer encoding technique was employed to identify the optimal category split, as recommended by the algorithm's developers. The METAR weather data, which is in a string format, was converted to a numerical score. This was done using the Air Traffic Management Airport Performance algorithm created by Eurocontrol [18]. This algorithm breaks down the raw METAR data into 5 sections - ceiling and visibility; wind; precipitation; freezing conditions; and dangerous phenomena - and assigns an integer score to each section. These scores are then added together to give a total score ranging from 0 to 30 where, the higher the number, the greater the severity of the weather conditions. In the case of the 'time of day' parameter, trigonometric (sine and cosine) coding was used to address the discontinuity occurring at midnight.

For each flight, the landing runway was considered to be the actual runway in use at the time of the prediction of the holding time. This was deduced from the latitude and longitude fields of the last message sent by the aircraft's transponder while airborne. This was easily done since Heathrow only has one active landing runway at a time. The information about the landing runway was then used to calculate the runway headwind and crosswind based on the wind speed and wind direction reported in the METAR.

Holding time can be affected by the Wake Turbulence Category (WTC) of landing aircraft. This is because ATCOs must take this parameter into account when sequencing aircraft on final, to ensure that a safe separation is maintained between consecutive landing aircraft. Runway capacity and efficiency are closely linked with the minimum separation between aircraft; for this reason EUROCONTROL has developed a recategorisation of the legacy ICAO wake turbulence scheme called RECAT-EU which has six categories: Super Heavy (CAT-A), Upper Heavy (CAT-B), Lower Heavy (CAT-C), Upper Medium (CAT-D), Lower Medium (CAT-E), and Light (CAT-F) [19]. For instance, CAT-B corresponds to aircraft such as the Airbus A330/A350 and Boeing B747/B777/B787, while CAT-D corresponds to aircraft such as the Airbus A220/A320, Boeing B737 and Embraer E-Jet E2 families, which are operated very frequently and have a high effect on traffic density at EGLL.

Another parameter that ATCOs need to take into account is the engine type (piston, jet or turboprop/shaft) of an aircraft, as this affects its power and approach speed which, in turn, affects the

minimum separation required between landing aircraft. The aircraft type for each flight was determined using the 'aircraft_data' method of the Traffic API [12], which works by matching the ICAO24 identifier from the raw data to a separate OSN database. The wake turbulence category (RECAT-EU) and the engine type for each aircraft are listed in DOC8643 which is issued by the International Civil Aviation Organisation (ICAO) [20]. An ICAO-produced Python script was used to download the data in this document and to determine the WTC and engine type for each flight in the dataset used in this study. For Dataset 1, these parameters were calculated for each flight as noted in Table 1. On the other hand, for Dataset 2, the number of aircraft in each WTC (or with a certain engine type) in a 15-minute interval was found, as explained in Table 2.

As can be observed from Table 1, several parameters were calculated at three different times, corresponding to the position of an aircraft at 0, 30 and 60 minutes from the boundary of the London TMA. On the other hand, as observed in Table 2, several parameters were calculated for each of the four holding stacks of EGLL.

Table 1. Dataset 1: Holding time of individual flights

Parameter	Type	Encoding[a]	Input/Output
Aircraft:			
-Flight Holding Time	Integer	-	Output
-Stack identifer	String	One hot	Input
-Wake Turbulence Category	String	One hot	Input
-Engine type	String	One hot	Input
-Ground Speed [b]	Integer	-	Input
-Track [b]	Integer	-	Input
-Altitude[b]	Integer	-	Input
Airport/holding conditions:			
-Weather Score[b]	Integer	-	Input
-Number of Holding Flights [b]	Integer	-	Input
-Current Landing Runway Identifier [b]	String	One hot	Input
-Landing Crosswind Component[b]	Integer	-	Input
-Landing Headwind Component [b]	Integer	-	Input
-Number of landings in the last hour [b]	Integer	-	Input
-Departure Delay index[b]	Integer	-	Input
-Arrival Delay Index [b]	Integer	-	Input
-Number of Canceled Flights [b]	Integer	-	Input
Time:			
-Day of the week	String	One hot	Input
-Time of day [b]	Cyclic	Trigonometric	Input

[a]LightGBM applies the Fisher [17] technique to find the optimal split over categories.

[b]This parameter was calculated three times: when the aircraft was at the boundary of the London TMA; 30 minutes before the TMA boundary; and 60 minutes before the TMA boundary.

Table 2. Dataset 2 : Average holding time of aircraft in a holding stack during a 15-minute time interval

Parameter	Type	Encoding[a]	Input/Output
Holding conditions:			
-Average holding time [b]	Integer	-	Output
-Number of aircraft in the holding stack [b]	Integer	-	Input
-Minimum holding time [b]	Integer	-	Input
-Maximum holding time [b]	Integer	-	Input
-Number of Aircraft for each Wake Turbulence Category	Integer	-	Input
-Number of Aircraft for each Engine Type	Integer	-	Input
-Number of aircraft which did not fly over any holding fix [c]	Integer	-	Input
Airport Conditions:			
-Number of landings in the last hour	Integer	-	Input
-Number of landings per runway during time interval	Integer	-	Input
-Weather Score	Integer	-	Input
-Landing Crosswind Component	Integer	-	
-Landing Headwind Component	Integer	-	Input
-Departure Delay Index	Integer	-	Input
-Arrival Delay Index	Integer	-	Input
-Number of Canceled Flights	Integer	-	Input
Time:			
-Day of the week	String	One hot	Input
-Start time of the 15-minute interval	Cyclic	Trigonometric	Input

[a]LightGBM applies the Fisher [17] technique to find the optimal split over categories.
[b]This parameter was calculated for each holding stack
[c]This includes aircraft that did not enter any holding pattern

3.4 Machine Learning Model Training

For this study, two different ML techniques were used to build models trained on the datasets presented in the previous section. These were LightGBM, which is a gradient boosting framework that uses tree-based learning algorithms, and the Classic (Vanilla) Stacked LSTM, a type of Recurrent Neural Network aimed at dealing with the vanishing gradient problem present in traditional RNNs.

Other suitable gradient-boosted decision trees methods, such as XGBoost or CatBoost, could have been used. In [21] and [22] different gradient-boosted decision trees methods, including LightGBM, were used to predict flight delays. The results of these studies indicated that LightGBM accelerates the traditional gradient-boosted decision trees training process by up to 20 times while exhibiting higher prediction accuracy. Consequently, LightGBM was selected as the gradient-boosted decision trees algorithm in this study.

GRU is a simpler alternative to LSTM but a recent study [23] in predicting the estimated arrival time during flight did not find significant differences in model performance between these two techniques. For this reason, GRUs were excluded from the study in [23]. In addition, LSTMs have shown better

performance when having a large dataset - as in the case of this study. Given the nature of the problem - with sequential data and long-term dependencies - it was decided to use LSTMs given their success in related studies and applications.

As can be observed from Table 3 , the problem of predicting holding times was framed in two different ways: (a) as a regression problem, where the objective is to predict the holding stack in the next 15 minutes of a specific flight and (b) as a time series regression problem, where the objective is to forecast the next average holding time in a holding stack in the next 15 minutes based on past values of holding time (and other independent variables). Both ML techniques - LightGBM and LSTM - were applied to each of these problems, as explained in the rest of this section.

As is standard for regression problems, Dataset 1 was randomly split such that 80% was allocated to training, while the remaining 20% was allocated to testing. In the case of the time series regression problem, Dataset 2 was also split such that 80% was used for training and 20% was used for testing. However, given the time series nature of the problem, the data was not split randomly because the ML models need to be trained on sequential data, and to prevent any data leakage. For this reason, the last two and a half months of data was reserved for testing.

Table 3. ML models trained for each type of problem

Type of problem	ML models	Dataset
Regression	Model 1: Predicts aircraft holding time when the aircraft is at the TMA boundary (input parameters correspond to the time when the aircraft is at the TMA boundary)	1
	Model 2: Predicts aircraft holding time when the aircraft is 30 minutes from the TMA boundary (input parameters correspond to the time when the aircraft is 30 minutes from the TMA boundary)	1
	Model 3: Predicts aircraft holding time when the aircraft is 60 minutes from the TMA boundary (input parameters correspond to the time when the aircraft is 60 minutes from the TMA boundary)	1
	Model 4: Predicts aircraft holding time when the aircraft is at the TMA boundary (input parameters include those captured at 0, 30 and 60 minutes from the TMA boundary)	1
Time Series Forecasting	Model 5: Forecasts the average holding time for the next 15 minute time interval at BIG[a]	2
	Model 6: Forecasts the average holding time for the next 15 minute time interval at OCK [a]	2
	Model 7: Forecasts the average holding time for the next 15 minute time interval at BNN [a]	2
	Model 8: Forecasts the average holding time for the next 15 minute time interval at LAM [a]	2

[a]Input Parameters Correspond to all Holding Stacks

3.4.1 LightGBM

LightGBM is a free API for Python and R created by Microsoft [24]. LightGBM was modified by Microsoft to achieve faster training speed, higher efficiency, lower memory usage, better accuracy, and the ability to handle large-scale data.

3.4.1.1 Regression

The hyperparameters of LightGBM were tuned using grid search. For this purpose, the Scikit-learn API 'GridSearchCV' method [25] was used to perform the tuning. This method works by exhaustively searching for an estimator over a specified grid of parameter values, and uses cross-validation to evaluate each combination of hyperparameters. Cross-validation splits the training dataset into multiple 'folds' and performs training and validation across these splits. The standard cross-validation of three folds was used for this study. Table 4 shows the hyperparameters included in the grid search, together with the range of values of each hyperparameter. The optimal hyperparameters are reported in Section 4.

Table 4. LightGBM hyperparameter tuning using grid search

Hyperparameter	Description	Grid Search Parameters
`num_leaves`	Number of leaves in each tree	15, 31, 50
`learning_rate`	Determines the step size at each iteration while moving towards a minimum of the loss function.	0.01, 0.05, 0.1
`n_estimators`	This is the number of boosting rounds, or the number of trees to be built	100, 200, 500
`feature_fraction`	Controls the proportion of features randomly chosen for each tree.	0.8, 0.9, 1.0 (80%, 90%, 100%)
`bagging_fraction`	The fraction of data to be used for each iteration, effectively controlling how much data is randomly sampled (without replacement) for each boosting round.	0.6, 0.8, 1 (60%, 80%, 100%)
`bagging_freq`	Determines how often bagging (subsampling) is performed during the training process.	1, 5, 10

3.4.1.2 Time Series Regression

LightGBM was applied to time series regression using the sktime API [26]. The 'ForecastingGridSearchCV' method was used to find the best window length (a multiple of 15 minutes) from the following values: 5, 10, 15, 20, 25 and 30 i.e. each window length of 1 corresponds to a 15-minute average. The recursive forecasting strategy was applied to LightGBM. With this approach, the ML model uses its own predicted values (of the average holding time in a holding stack) to make predictions for future time steps, making it suitable for multistep forecasting.

3.4.2 LSTM

3.4.2.1 Regression

For the LSTM the Tensorflow Keras API [27] was used to create the ML models. For each input parameter, PowerTransformer from sklearn API [25] was applied as a preprocessing technique together with standardization (mean=0, std=1). The window length was set to one time step. GridSearchCV was used to find the best hyperparameters, as shown in Table 5. The stacked, dropout-regularized, tuned LSTM model having the ReLU activation function was implemented. This involves having a model composed of 4 layers: the 2 LSTM layers, 1 dropout layer, and 1 dense layer, which produces the final prediction. The neurons in each LSTM layer capture complex temporal patterns and relationships between different time steps in the data.

Table 5. LSTM Regression Hyperparameter Tuning Using Grid Search

Hyperparameter	Description	Grid Search Parameters
`Number of Units`	The number of neurons	64, 128
`batch_size`	The number of samples processed before the model updates its weights.	32,64
`epochs`	The model updates its weights once after every batch and processes all batches in one epoch.	50,100
`optimizer`	The algorithm that adjusts the model's weights to minimize the loss function. It determines how the model learns.	'adam ', 'adadelta'
`learning_rate`	Controls how much to change the model in response to the estimated error each time the model weights are updated	0.01,0.05
`dropout`	A regularization technique, in which some proportion of the neurons in a network are ignored to prevent overfitting	0.2,0.6(20%, 60%)

3.4.2.2 Time Series Regression

For the LSTM time series regression problem, similar configuration parameters (Vanilla architecture with 50 neurons for each layer, activation function, etc.) were applied as for the regression problem, together with the normalisation of the data using the 'MinMaxScaler' method [2]. The 'GridSearchCV' method was applied to find optimal values for batch size, number of epochs, and type of optimizer as shown in Table 6. The window length was set to 30 (7.5 hours).

Table 6. LSTM Forecasting Hyperparameter Tuning Using Grid Search

Hyperparameter	Description	Grid Search Parameters
`batch_size`	The number of samples processed before the model updates its weights.	16, 32
`epochs`	The model updates its weights once after every batch and processes all batches in one epoch.	8, 10
`optimizer`	The algorithm that adjusts the model's weights to minimize the loss function. It determines how the model learns.	'adam', 'adadelta'
`learning_rate`	Controls how much to change the model in response to the estimated error each time the model weights are updated	0.01, 0.05, 0.1
`dropout`	A regularization technique, in which some proportion of the neurons in a network are ignored to prevent overfitting	0.2, 0.4, 0.6 (20%, 40%, 60%)

[2] MinMax scaling resulted in superior predictions compared to the PowerTransformer scaling technique.

4. Results and Discussion

This section presents and analyses the results obtained during the training and testing of the ML models described in the previous section.

4.1 Regression

The performance of the ML models was determined using the test datasets, and measured in terms of Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). These metrics are shown in Tables 7 and 8 for Models 1-4, together with the LightGBM and LSTM hyperparameters. From Table 7 it can be observed that Models 1-4 have very similar LightGBM hyperparameters, with minor variations in the feature_fraction and bagging_fraction. It can also be seen that the RMSE and MAE both increase as the aircraft gets further away from the TMA boundary, as in the case of Models 2 and 3. This is expected since Model 1 has the most recent data with which to predict aircraft holding time while, in the case of Models 2 and 3, the uncertainty of the data is higher. The best predictive performance is obtained with Model 4 which slightly outperforms Model 1. This is because Model 4 was trained with all of the training data. Figure 6 shows a histogram of the time difference between the predicted and actual aircraft holding times obtained for Model 4 with the test dataset. This histogram shows that the time difference follows a normal distribution, with a mean of -0.02 minutes and a standard deviation of 2.23 minutes.

From Table 8 it can be observed that there are minor differences between the LSTM hyperparameters - determined through hyperparameter tuning, as explained in Section 4 - of Models 1-4. The greatest accuracy is obtained with the largest number of neurons (128) - resulting in longer training times - and epochs (100) - meaning that more iterations over Dataset 1 were required to minimise loss. It can also be observed that, as in the case of LightGBM, the best predictive performance with LSTM is obtained with Model 4. However, the RMSE and MAE of the models trained with LSTM are higher than those of the corresponding models trained with LightGBM.

A SHAP impact analysis was performed on each regression model trained with LightGBM and LSTM. The SHAP beeswarm plots in Figures 7-10 summarise how different features impact model predictions. Each point represents a sample, with its colour indicating the original value of the feature (blue for low, red for high) and its position on the x-axis indicating the SHAP value, meaning how much that feature contributes to increasing or decreasing the model output (i.e. the predicted holding time in this case). In each plot, the top 10 features are shown and these are ranked by their mean absolute SHAP value.

From Figures 7 - 10 it can be observed that the importance of different features varies from one model to another. However, in the majority of cases, the most important feature is the number of holding flights (i.e. the size of the queue). As expected, when the value of this parameter increases, it tends to have a bigger positive impact on the output i.e. the predicted holding time increases. For Model 2 (LSTM) and Model 3 (LightGBM), the sine component of the time of day is the most important feature, with lower values tending to have a negative impact on the output, and higher values tending to have a positive impact. In the case of Model 3 (LSTM), the features corresponding to the holding stack identifiers have the biggest impact on the output, with the biggest (negative) impact corresponding to the BNN holding stack. Figure 9 (right) suggests that when the BNN holding stack is used, the holding time of a flight is less than when any of the other holding stacks is used. The stack identifier is also one of the top 5 features of the LightGBM models.

The day of the week is also an important feature of Model 3 (LSTM), with the biggest (negative) impact on the output occurring when a flight is on a Wednesday or, to a lesser extent, on a Thursday. This may be due to the fact that traffic into Heathrow tends to be at its lowest towards the middle of the week.

Table 7. Performance Metrics and Hyperparameter of the regression models trained with LightGBM

Model Number	1 (TMA)	2 (TMA-30)	3 (TMA-60)	4 (All data)
RMSE (Minutes)	2.30	2.69	2.78	2.25
MAE (Minutes)	1.53	1.92	2.0	1.50
num_leaves	50	50	50	50
learning_rate	0.1	0.1	0.1	0.1
n_estimators	500	500	500	500
feature_fraction	0.9	1.0	1.0	0.8
bagging_fraction	1	1.0	0.8	0.8
bagging_freq	1	1	1	1

Table 8. Performance metrics and hyperparameters of the regression models trained with LSTM

Model Number	1 (TMA)	2 (TMA-30)	3 (TMA-60)	4 (All Data)
RMSE (Minutes)	2.90	3.31	3.45	2.88
MAE (Minutes)	2.07	2.54	2.61	2.07
Number of Units	128	128	128	128
batch_size	32	64	32	32
epochs	100	100	100	100
optimizer	adadelta	adam	adam	adadelta
learning_rate [a]	-	0.01	0.01	-
dropout	0.2	0.2	0.2	0.2

[a]The 'adadelta' optimizer does not rely on the learning rate since it is designed to elimate the need for manual tuning.

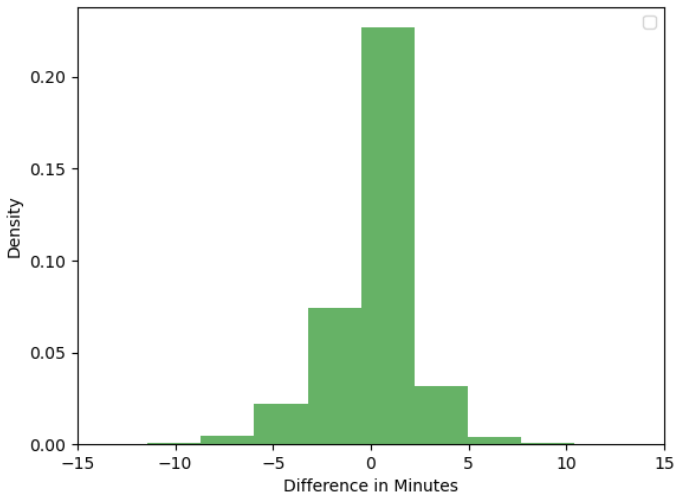

Figure 6. Distribution of time difference between actual and predicted aircraft holding times (Model 4 with LightGBM)

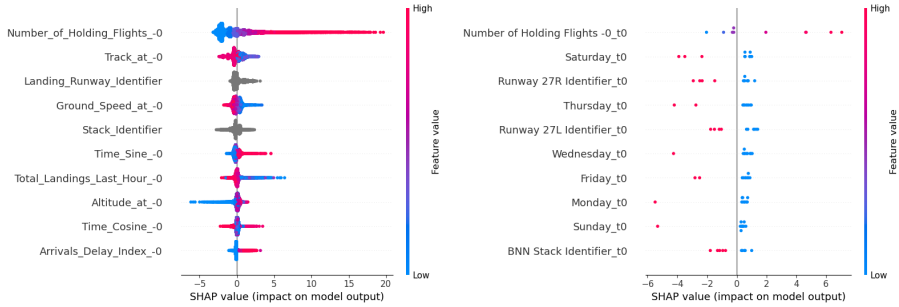

Figure 7. SHAP summary plot of Model 1 for LightGBM (left) and LSTM (right).

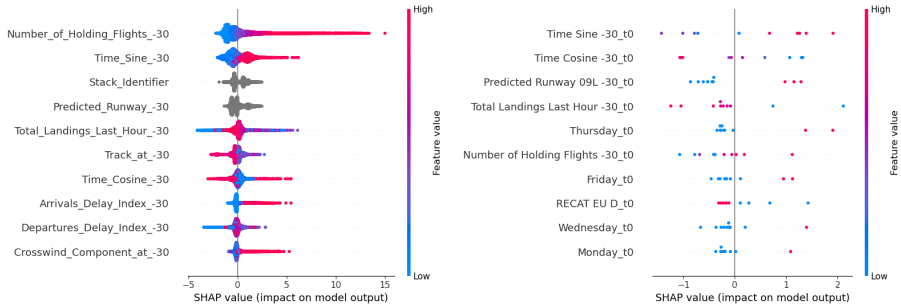

Figure 8. SHAP summary plot of Model 2 for LightGBM (left) and LSTM (right).

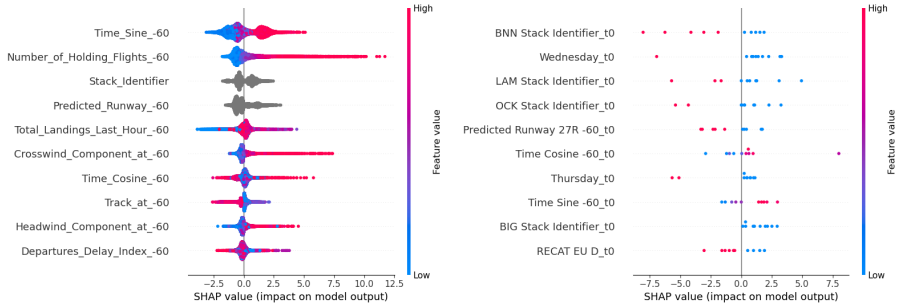

Figure 9. SHAP summary plot of Model 3 for LightGBM (left) and LSTM (right).

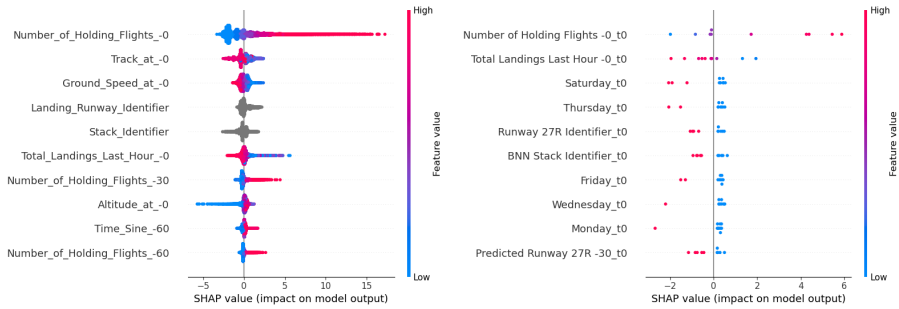

Figure 10. SHAP summary plot of Model 4 for LightGBM (left) and LSTM (right).

4.2 Time Series Forecasting

The results of the time series regression models trained with LightGBM and LSTM are shown in Tables 9 and 10 respectively.

From Table 9 it can be observed that, as a result of grid search, Models 5, 7 and 8 adopted a window length of 7 hours and 30 minutes (30 x 15 minutes), and Model 6 adopted a window length of 6 hours and 15 minutes. The optimal hyperparameters of the forecasting models trained with LSTM are shown in Table 10 and it can be seen that these are different for each model. For instance, in the case of Models 5 and 6, a reduced batch size was required before the model weights were updated. Models 6, 7 and 8 handled all batch processes with a minimum epoch count of 8. While Models 5, 6 and 8 used the 'adam' optimiser (with a learning rate of 0.01), the other models used the 'adadelta' optimiser.

Model 7 (BNN) has the worst RMSE and MAE of Models 5-8, both in the case of LightGBM and LSTM. This could be because there were fewer flights in the BNN holding stack (than in the other stacks) between July and March, and this could have affected the model's training. Model performance could be improved, for instance, by using a larger dataset and/or the Synthetic Minority Over-sampling Techniques (SMOTE) such as Dynamic Time Warping-SMOTE or Temporal-orientated-SMOTE.

Figures 11, 12, 13, and 14 show the SHAP beeswarm plots for Models 5, 6, 7 and 8, respectively. In each plot, the top ten features are shown and these are ranked by their mean absolute SHAP value. In the case of LightGBM, it can be observed that the most impactful feature of Models 5, 7 and 8 is the total number of landings in the previous hour. When the number of landings increases, the average predicted holding time also tends to increase. This is probably because the airport is busier - and closer to its operational capacity - and more aircraft have to hold for a longer time. For Model 6 (with LightGBM) the most impactful feature is the number of aircraft in the holding stack with a jet engine. When the number of aircraft with a jet engine increases, so does the average holding time. Jet aircraft account for the vast majority of traffic into Heathrow; therefore, when their number increases, the airport is busier and more aircraft end up holding.

In the case of LSTM, it can be observed that the average predicted holding time in a holding stack is impacted by the holding time characteristics of other stacks. For instance, for Model 5, the holding time at LAM has an impact on the holding time at BIG. Basically, when the holding time at LAM increases, the average holding time at BIG tends to decrease (note, however, that this does not imply causation). Similarly, for Model 6, the holding time at LAM has an impact on the holding time at OCK; when the holding time at LAM decreases, the holding time at OCK tends to decrease slightly. For Model 7, the number of landings on Runway 09L has the most impact on the predicted average holding time of BNN - when the number of landings on Runway 09L is small, the holding time of BNN is reduced. The number of landings on Runway 27L also has an impact on the predicted average holding time of BNN, but not to the same extent as Runway 09L. This may be due to the fact that specific holding stacks are normally used in conjunction with specific arrival routes and runways. For Model 8, the most impactful features on the average holding time at LAM are the holding time characteristics of LAM itself. For instance, a low holding time at LAM (over the previous 7.5 hours) tends to have a negative impact on the predicted holding time in the next 15 minutes (and vice-versa).

The forecasting models trained with LSTM have lower average error metrics than those trained with LightGBM. With LSTM, the average RMSE and MAE (over all holding stacks) decreased by 37.1% and 41.7%, respectively, when compared to LightGBM. The best results were obtained with LSTM Model 8. Figure 15 shows a 6 hours rolling average holding time plot of the results obtained when Model 8 was tested on the test dataset (the final 2 and a half months of data). A window length of 7 hours and 30 minutes was used. The overlap between the original and predicted average holding times in LAM can be seen in the plot; however, the model had difficulty to predict the outliers with long

average holding times.

Table 9. Performance metrics and window length of the time series regression models trained with LightGBM

Model Number	5 (BIG)	6 (OCK)	7 (BNN)	8 (LAM)	Average Error (Minutes)
RMSE (Minutes)	3.94	3.63	4.47	3.86	3.98
MAE (Minutes)	2.45	2.23	3.58	2.36	2.66
Window Length[a]	30	25	30	30	-

[a]Window Length is a multiple of 15 minutes

Table 10. Performance metrics and hyperparameters of the time series regression models trained with LSTM

Model Number	5 (BIG)	6 (OCK)	7 (BNN)	8 (LAM)	Average Error (Minutes)
RMSE (Minutes)	2.57	2.45	2.45	2.17	2.41
MAE (Minutes)	1.57	1.48	1.63	1.20	1.47
batch_size	16	16	32	32	-
epochs	10	8	8	8	-
optimizer	adam	adam	adadelta	adam	-
learning_rate[a]	0.01	0.01	-	0.01	-
dropout	0.4	0.6	0.4	0.4	-
Window Length[b]	30	30	30	30	-

[a]The adadelta optimiser does not rely on the learning rate since it is designed to elimate the need for manual tuning.
[b]Window Length is a multiple of 15 minutes

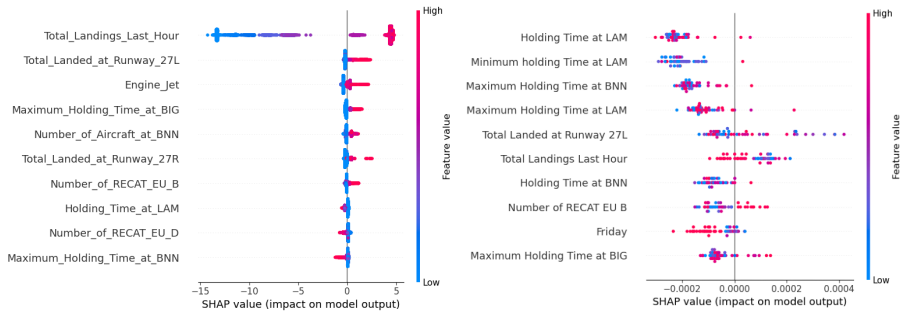

Figure 11. SHAP summary plot of Model 5 for LightGBM (left) and LSTM (right).

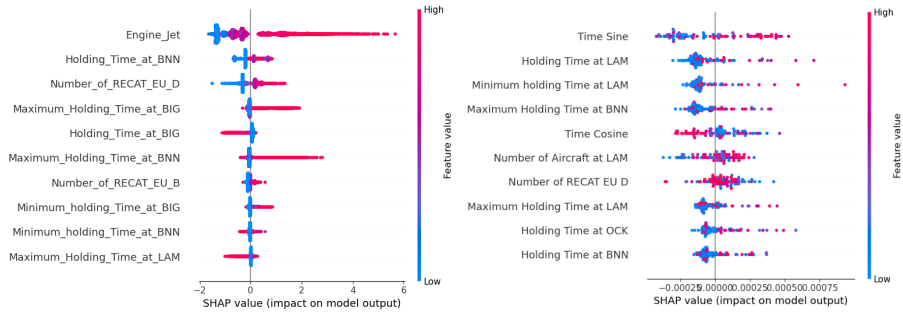

Figure 12. SHAP summary plot of Model 6 for LightGBM (left) and LSTM (right).

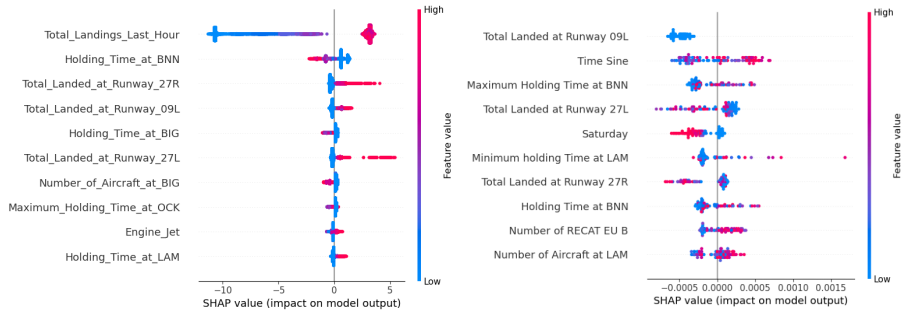

Figure 13. SHAP summary plot of Model 7 for LightGBM (left) and LSTM (right).

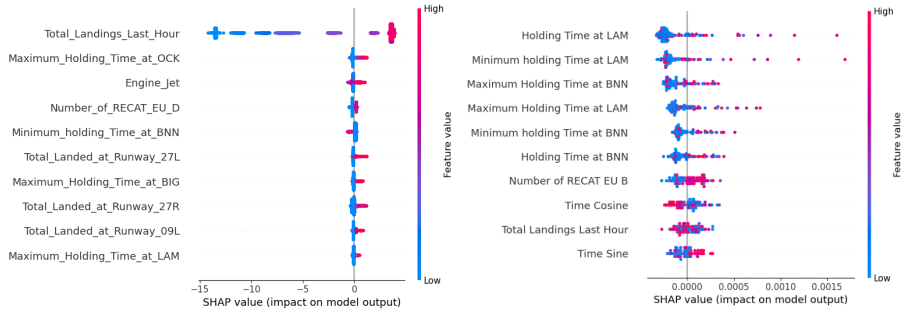

Figure 14. SHAP summary plot of Model 8 for LightGBM (left) and LSTM (right).

5. Conclusion and Future Work

This study applied two ML techniques - LightGBM and classic LSTM - to train ML models to predict holding times at London Heathrow Airport using one year of historical data. Two sets of models were trained: one set of regression models to predict the holding time of individual flights at different times from the London TMA; and one set of time series regression models to predict the average holding time in each holding stack. Test results of the regression models showed that the models trained with LightGBM had the best performance, with minimum RMSE and MAE values of 2.25 and 1.50 minutes, respectively. On the other hand, the results of the time series regression models showed better performance by the models trained with LSTM, with average RMSE and MAE values of 2.41 and 1.47 minutes, respectively. These results demonstrate the suitability of the selected ML

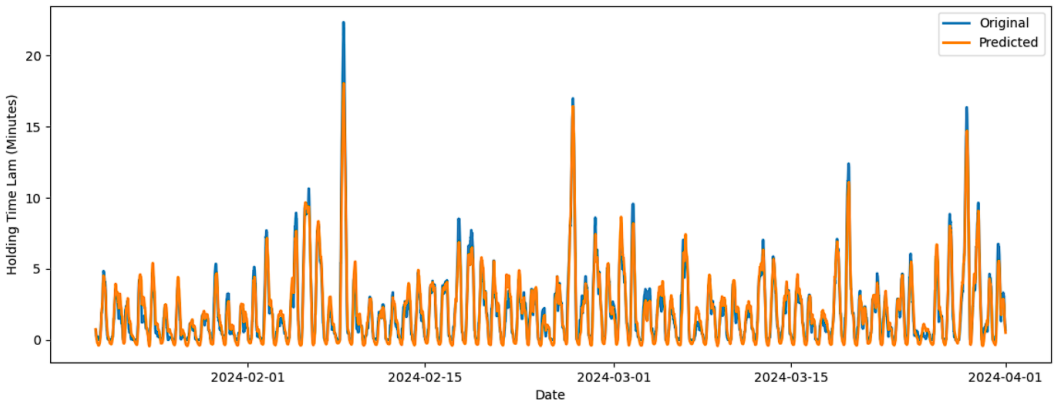

Figure 15. Original and Predicted 6-hour rolling average holding times of LAM with LSTM Model 8

techniques for holding time prediction.

For these ML models to be deployed in an operational Air Traffic Management environment, they would need to be integrated with existing ATC systems and be able to receive real-time data (such as Departure Delay Index data, surveillance data, and weather reports) from various sources. The models could be integrated, for instance, with the Arrival Manager (or Extended Arrival Manager) currently deployed at EGLL to optimise aircraft sequencing and reduce TMA delays.

In the case of Models 1-4, the aircraft holding time is only predicted up to 1 hour in advance. These predictions could be extended to several hours, even to the take-off time of a particular flight. This would be more operationally useful from the pilots' perspective, but would come at the expense of increased prediction uncertainty. It would also require the creation of new regression models, e.g. a model to predict holding time when an aircraft is 2 hours from the TMA, etc. In the case of Models 5-8, the models can be used 'as is' to forecast the holding time at any point in the future in 15-minute intervals. Thus, they could be used by controllers, for instance, to forecast average holding time in a particular holding stack over the next 24 hours.

Several avenues for future work are identified, including the addition of new features (such as the weather conditions of surrounding airports; and airport capacity metrics) and the exploration of other ML techniques (such as GRUs and CatBoost) and alternative encoding methods (such as embeddings). This study also identified the necessity of developing an ML model capable of predicting an aircraft's landing runway. Furthermore, to reduce the number of models and improve generalisation, work can be done to develop an ML model that can forecast the mean delay associated with each holding stack in a multi-output framework, or a model that integrates different lookahead times.

Author contributions

- Michele Vella: Data acquisition and curation, coding and testing, analysis, visualization, writing (original draft)
- Jason Gauci: Conceptualization, methodology, supervision, writing (review and editing)
- Alexiei Dingli: Supervision, writing (review)

Open data statement

The datasets created for these studies are provided at:
https://github.com/MICHELEVELLA/Data-Driven-Prediction-of-Aircraft-Holding-Times-Using-OpenSky-Data-The-12th-OpenSky-Symposium-.

Reproducibility statement

Source code related to the study can be found at:
https://github.com/MICHELEVELLA/Data-Driven-Prediction-of-Aircraft-Holding-Times-Using-OpenSky-Data-The-12th-OpenSky-Symposium-

References

[1] Iacopo Prissinotti. *EUROCONTROL calling for common action to address air traffic control delays*. https://www.eurocontrol.int/press-release/eurocontrol-calling-common-action-address-air-traffic-control-delays. 2024.

[2] EUROCONTROL. *Point Merge Improving and harmonising arrival operations*. https://www.eurocontrol.int/concept/point-merge. 2024.

[3] Junzi Sun Xavier Olive. "Environmental inefficiencies for arrival flights at European airports". In: *Plos One* (2023).

[4] Anibal Bregon Jorge Silvestre Miguel A. Martínez-Prieto. "A deep learning-based approach for predicting in-flight estimated time of arrival". In: *The Journal of Supercomputing 80:17212–17246* (2024).

[5] C. Cetek O. Basturk. "Prediction of aircraft estimated time of arrival using machine learning methods". In: *The Aeronautical Journal 2021;125(1289):1245-1259* (2021).

[6] Sim Kuan Goh Imen Dhief Zhi Jun Lim. "Speed Control Strategies for E-AMAN using Holding Detection-Delay Prediction Model". In: *10th SESAR Innovation Days* (2020).

[7] Imen Dhief Lim Zhi Jun Sameer Alam. "Towards a greener Extended-Arrival Manager in air traffic control: A heuristic approach for dynamic speed control using machine-learned delay prediction model". In: *Journal of Air Transport Management 103* (2022).

[8] Yicheng Zhang Liping Huang Sheng Zhang. "Aircraft Landing Time Prediction with Deep Learning on Trajectory Images". In: *13th SESAR Innovation Days* (2023).

[9] Gabriel Jarry Ramon Dalmau Philippe Very. "On the Causes and Environmental Impact of Airborne Holdings at Major European Airports". In: *Journal of Open Aviation Science* (2024).

[10] Michael Bloem Heather Arneson. "A Method for Scheduling Air Traffic with Uncertain En Route Capacity Constraints". In: *AIAA Guidance, Navigation, and Control Conference* (2009).

[11] The OpenSky Network. *Open Air Traffic Data for Research*. https://opensky-network.org/. [Online; accessed 01-October-2024]. 2024.

[12] Xavier Olive. *air trafic data processing with Python*. https://traffic-viz.github.io/index.html. [Online; accessed 01-October-2024]. 2022.

[13] Iowa Environmental Mesonet. *ASOS-AWOS-METAR Data Download*. https://mesonet.agron.iastate.edu/request/download.phtml. [Online; accessed 01-October-2024]. 2024.

[14] AeDBX. *AeroDataBox*. https://rapidapi.com/aedbx-aedbx/api/aerodatabox. [Online; accessed 01-October-2024]. 2024.

[15] NATS. *eAIS Package United Kingdom*. https://www.aurora.nats.co.uk/htmlAIP/Publications/2025-04-17-AIRAC/html/index-en-GB.html. [Online; accessed 20-April-2025]. 2025.

[16] X. Olive, L. Basora, J. Sun, and E. Spinielli. "A deep learning-based approach for predicting in-flight estimated time of arrival". In: *Journal of Open Aviation Science* 2 (Feb. 2025). DOI: https://doi.org/10.59490/joas.2024.7943.

[17] Walter D. Fisher. "On Grouping for Maximum Homogeneity". In: *Journal of the American Statistical Association* 53(284) (1958). DOI: https://doi.org/10.1080/01621459.1958.10501479.

[18] Eurocontrol. *Algorithm to describe weather conditions at European airports*. https://www.eurocontrol.int/sites/default/files/publication/files/algorithm-met-technical-note.pdf. [Online; accessed 01-October-2024]. 2011.

[19] Eurocontrol. *"RECAT-EU" European Wake Turbulence Categorisation and Separation Minima on Approach and Departure*. https://www.eurocontrol.int/archive_download/all/node/9681. [Online; accessed 20-January-2025]. 2024.

[20] ICAO (International Civil Aviation Organization). *Aircraft Type Designators (Doc 8643/52)*. https://store.icao.int/en/aircraft-type-designators-doc-8643. [Online; accessed 01-October-2024]. 2024.

[21] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: Dec. 2017.

[22] Jiang Tao, Hua Man, and Li Yanling. "Flight delay prediction based on LightGBM". In: *2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*. 2021, pp. 1248–1251. DOI: 10.1109/ICCASIT53235.2021.9633431.

[23] Jorge Silvestre, Miguel A. Martínez-Prieto, Anibal Bregon, and Pedro Álvarez-Esteban. "A deep learning-based approach for predicting in-flight estimated time of arrival". In: *The Journal of Supercomputing* 80 (Apr. 2024), pp. 1–35. DOI: 10.1007/s11227-024-06060-6.

[24] Microsoft Corporation. *Welcome to LightGBM's documentation*. https://lightgbm.readthedocs.io/en/stable/. [Online; accessed 01-October-2024]. 2023.

[25] Scikit-learn. *Machine Learning in Python*. https://scikit-learn.org/stable/. [Online; accessed 01-October-2024]. 2024.

[26] Sktime. *Welcome to sktime*. https://www.sktime.net/en/stable/. [Online; accessed 01-October-2024]. 2024.

[27] TensorFlow. *Keras: The high-level API for TensorFlow*. https://www.tensorflow.org/guide/keras. [Online; accessed 01-October-2024]. 2024.