Journal of Open Aviation Science

JOAS

ARTICLE (*Pre-print*)

Open Machine Learning Models for Actual Takeoff Weight Prediction

First Author [*,1] Second Author [2] and Third Author[3,1]

[1]Institution-1, City, Country
[2]Institution-2, City, Country
[3]Institution-3, City, Country
*Corresponding author: correspondence@email.domain

Abstract

This study uses large-scale open aviation data to develop an open-source machine learning model to accurately predict commercial flights' actual takeoff weight (ATOW). Given the significance of accurately estimating ATW for environmental impact assessments, this research utilizes advanced ensemble learning techniques—CatBoost, LightGBM, and XGBoost—known for their robust performance in structured data analysis and high predictive accuracy. The motivation stems from the Eurocontrol Performance Review Commission (PRC) Data Challenge, which advocates for transparent models to enhance studies related to aviation's influence on climate change. The models are evaluated based on their efficiency, accuracy, and applicability to real-world data, aiming to contribute significantly to aviation analytics and environmental research.

Keywords: keyword; keyword-two; keyword number three

Abbreviations: JOAS: Journal of Open Aviation Science, ATM: Air Traffic Management

1. Introduction

Air Traffic Management (ATM) advancements are crucial for achieving a globally sustainable aviation industry. The implementation of novel technologies and operational concepts, such as Trajectory-Based Operations (TBO), have been explored worldwide to reach increasingly stringent environmental performance targets [1]. This requires advanced tools for modeling and predicting flight trajectory performance and its associated ecological impact across multiple scales. To this goal, extensive previous work has explored analytical and empirical approaches for trajectory prediction [2, 3], fuel burn estimation [4, 5, 6] and emissions assessment [7, 8], considering the different phases of the flight operation.

As a fundamental parameter in flight dynamics, aircraft weight is an essential input for these models. However, this information is only available in detailed aircraft data registered by Flight Data Recorder (FDR) systems, being considered proprietary and kept confidential by airlines. This lack of openly available data is typically compensated with assumptions and estimations of aircraft mass, which might introduce a significant source of error and lead to inaccurate trajectory predictions [9].

Previous studies have focused on improving mass estimations with the use of other sources of data, such as aircraft surveillance data. Alligier et al. [10] introduced a least squares method to estimate the mass from past trajectory points using the physical model of the aircraft and radar data. With the advent of Automatic Dependent Surveillance-Broadcast (ADS-B), flight trajectory data has become increasingly accessible to the general public through flight tracking service providers such as the OpenSky Network [11], allowing for novel data-driven modeling approaches to be explored by researchers. Taking advantage of large-scale historical ADS-B data from the OpenSky Network,

Alligier [12] developed a machine learning approach, using neural networks and gradient boosting machines to create predictive models of the mass and speed profile during the climb phase.

Specifically focused on the takeoff phase, Sun et al. [13] used ADS-B data in combination with physical kinetic models to infer aircraft takeoff weight based on two different estimation methods. More recently, the authors [14] developed a Bayesian inference approach to estimate the initial mass at takeoff, using independent aircraft mass estimates at different flight phases computed with analytical models that incorporate trajectory information. However, these approaches still relied on prior knowledge of aircraft dynamics and aircraft performance parameters that are not openly available.

This work aims to develop an open-source machine learning model to predict the actual takeoff weight of a flight based on large-scale open aviation data. The machine learning methods applied in this study include *LightGBM*, *AdaBoost*, and *XGBoost*, selected for their effectiveness in handling structured data and ability to deliver high predictive accuracy. The work has been mainly motivated by the Eurocontrol Performance Review Commission (PRC) Data Challenge [15], which emphasized the need for open models to support studies assessing the impact of aviation on climate change.

The remainder of this article is organized as follows. Section 2 presents the methodology used to process the data and develop the predictive models. In Section 3, we discuss the results, comparing and analyzing different ML models. Finally, Section 4 concludes and shares ideas for future work.

2. Method

This section provides a comprehensive overview of the methods used in this study, including data preprocessing, feature engineering, and predictive modeling techniques. Additionally, we describe the datasets and the steps taken to prepare and enhance them, focusing on how meaningful features were extracted to improve model performance.

2.1 Data Description

This subsection details the open-source datasets used in the study, including flight and trajectory data, as well as supplementary sources that enrich the analysis. These datasets provide essential information on flights, aircraft characteristics, and operational parameters, serving as the basis for building predictive models.

2.1.1 Flight and Trajectory Data

The main data used in this work for predictive modeling of the actual takeoff weight of a flight is provided by Eurocontrol through the PRC data challenge.

The challenge data is composed of two datasets. The first one contains flight data for a set of 369,013 flights that departed or arrived at a European airport in 2022. It provides the flight identification, origin and destination airports, aircraft type and wake turbulence category, airline as well as operational parameters such as actual departure and arrival times, flight duration, taxi-out time, flown distance, and the actual takeoff weight (ATOW).

The second dataset provides detailed trajectory data for each flight, including 4D position reports (longitude, latitude, altitude, timestamp), ground speed, track angle, and vertical climb/descent rates. For example, Figure 1 shows the horizontal profile of flight trajectories for one day of operations in the European airspace. Additionally, when available, the dataset provides meteorological data for each 4D position, encompassing the wind components and temperature. It is worth mentioning that flight trajectories are not necessarily complete because of limited ADS-B coverage in some parts of the world and lower altitudes.

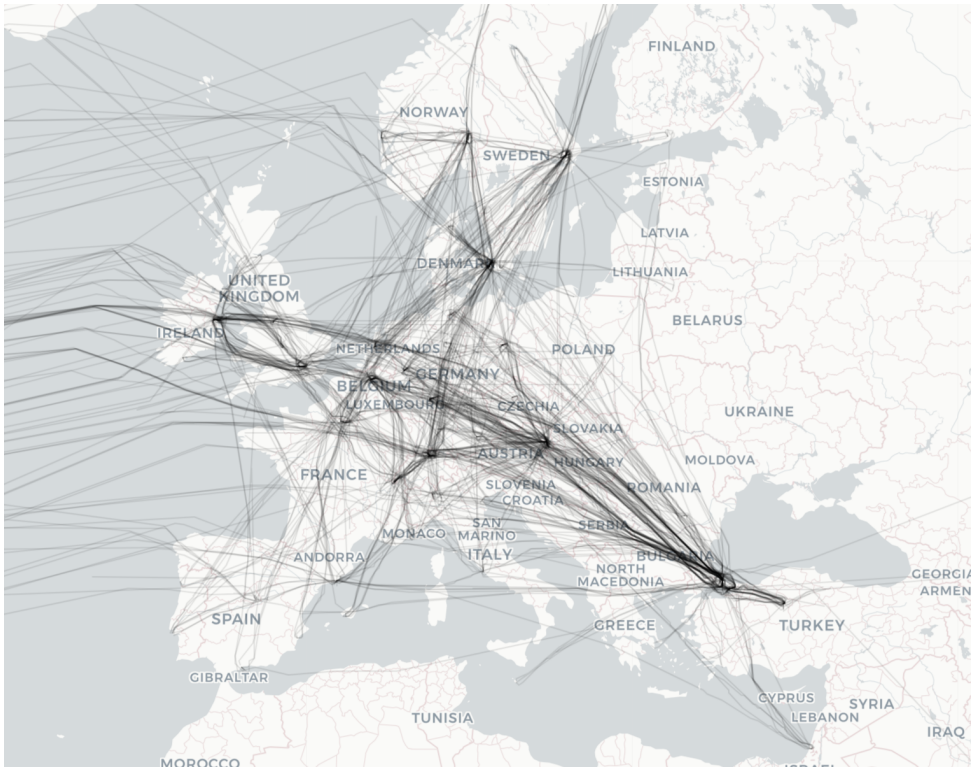

Figure 1. Horizontal profile of trajectories flown in the European airspace on April 2, 2022.

2.1.2 Supplementary Data

This study utilized two open-source supplementary datasets to gather airports' geographic coordinates (latitude, longitude) and altitude: the OpenFlights Airports Database and The Global Airport Database. While these sources provided extensive coverage, 15 airports were still missing in both the challenge and submission sets, which we manually added to the dataset to ensure completeness. Additionally, we incorporated an open-source dataset from the Federal Aviation Administration (FAA), which contains detailed aircraft characteristics. This dataset, updated in September 2023, ensures that we use the most current specifications available during the study, providing accurate aircraft performance metrics and operational characteristics for our analysis. Furthermore, we created a supplementary database to capture additional aircraft features not present in the FAA dataset. This additional data was compiled manually from public sources, including Wikipedia, technical sheets, factory specifications, and Eurocontrol Data, to enhance the accuracy and completeness of our analysis. For more information on the datasets, refer to the source code of this work.

2.2 Data Preprocessing

In this study, trajectory data preprocessing consisted of two main steps: merging datasets and segmenting the flight trajectory into three phases – departure, en route, and arrival – using 100 NM (nautical mile) cylindrical regions. The trajectory data and other flight information were integrated to represent flight trajectories comprehensively. The merging process involved aligning the datasets based on unique flight identifiers and timestamps. This step was essential for ensuring consistency in each flight record's temporal and spatial attributes, enabling accurate tracking of aircraft positions and speeds over time. In Figure 2, there is an illustration of this process. Also, to analyze the

various phases of flight, each trajectory was segmented into three key stages: departure, en route, and arrival. This segmentation was based on the distance from the origin and destination airports using 100 NM cylindrical regions as reference points.

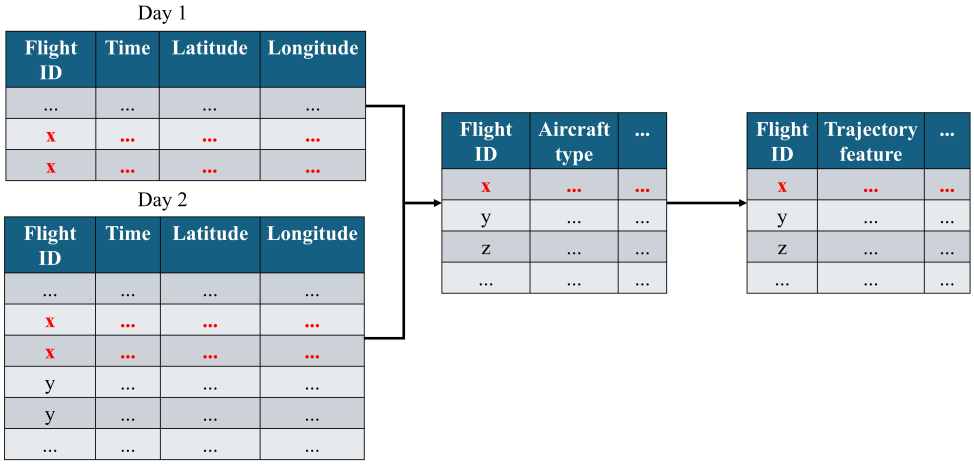

Figure 2. Merging process.

Thus, the departure phase is defined as the segment of the flight trajectory from takeoff until the aircraft exits a 100 NM radius cylinder centered on the departure airport, capturing the aircraft's initial climb and acceleration. The en-route phase represents the flight segment between the departure and arrival cylinders, characterized by the aircraft's cruise altitude and relatively stable speed and heading. The arrival phase corresponds to the flight segment from when the aircraft enters the 100 NM radius cylinder centered on the destination airport until it lands, encompassing the descent and approach procedures. The segmentation was implemented using a spatial filtering technique that evaluates the aircraft distance from the respective airports, allowing for precise identification of entry and exit points into the 100 NM zones. This approach ensures that each flight phase is accurately identified, facilitating subsequent analyses of the aircraft behavior in different flight stages. It is represented in Figure 3.

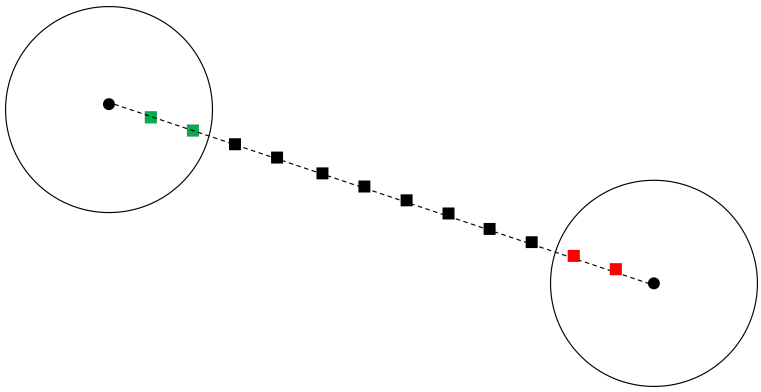

Figure 3. Segmentation process.

2.3 Feature Engineering

This subsection details the feature engineering techniques applied to preprocess and enhance the dataset for modeling purposes. The goal was to extract meaningful features that capture temporal patterns, flight characteristics, geographical information, spatial relationships, and interactions between various flight parameters.

2.3.1 Flight Data

We engineered several new features from the flight data to capture temporal dynamics, flight efficiency, geographical context, and the relationship between different flight parameters. Specifically, we focused on temporal attributes extracted from departure and arrival times, duration-based metrics like taxi ratios and flight speeds, geographical categorizations based on regions and countries, interaction terms combining airspeed and specific energy, and ratio calculations between vertical rate and airspeed.

Temporal Feature Extraction From the departure time, we extracted attributes such as the hour and minute of departure, day of the week, month, week of the year, and the season—categorized into Winter, Spring, Summer, or Fall based on the month. This allowed us to capture daily, weekly, and seasonal patterns that could influence flight operations. For example, flights departing during peak hours or specific seasons might experience different conditions affecting performance. Additionally, we created boolean flags based on the departure time to indicate whether the flight departed on a weekend or during rush hours (defined as 7–9 AM or 4–6 PM). These features help assess the impact of higher traffic volumes on flight performance.

We extracted the hour and minute of arrival from the arrival time, enabling us to analyze temporal patterns at the destination. We calculated the duration between the off-block time and arrival time in minutes, accurately measuring the total flight duration.

Duration-Based Feature Creation We engineered several duration-based features to capture aspects of flight efficiency and performance. We calculated the ratio of taxi-out time to flight duration, known as the taxi ratio, which assesses the proportion of time spent taxiing relative to the total flight duration. A high taxi ratio might indicate congestion at the airport or inefficiencies in ground operations. We normalized the taxi ratio by dividing it by the median, facilitating comparison across flights.

We also calculated the average speed of the flight by dividing the flown distance by the flight duration. This feature helps analyze operational efficiency over different distances and durations. To maintain data consistency, missing values in flight speed were filled with the median.

To categorize flights based on duration, we introduced a flight duration category by binning the flight durations into Very Short (0–60 minutes), Short (60–180 minutes), Medium (180–300 minutes), and Long (over 300 minutes). This categorization aids in segmenting flights for analysis and modeling purposes.

Additionally, we computed the speed per distance by dividing the flight speed by the flown distance. This feature provides a normalized speed metric relative to the flight distance, assisting in analyzing speed efficiency over varying distances.

Geographical Feature Engineering To incorporate geographical context and spatial relationships, we calculated the distance between the departure and arrival airports using the Haversine

formula, which computes the great-circle distance between two points on the Earth's surface based on their latitudes and longitudes. This resulted in a new feature representing the actual distance of the flight path. We also calculated the altitude difference between the arrival and departure airports to capture the vertical distance traversed during the flight. The bearing between the two airports was computed to determine the initial compass direction from the departure airport to the arrival airport. Additionally, we calculated the elevation gradient by dividing the altitude difference by the actual distance, representing the altitude change rate per kilometer. These features can influence fuel consumption and flight performance.

Furthermore, we performed clustering based on the departure and arrival airports' geographical coordinates (latitude and longitude). By combining these coordinates and applying the K-Means clustering algorithm, we aimed to identify spatial groupings of airports. We also grouped airports by region based on their country codes, assigning the departure and arrival airports to regions such as Europe, North America, South America, the Middle East, Asia, Africa, and others. This regional classification facilitates the analysis of patterns and differences in flight operations across regions. We developed features to indicate whether a flight was domestic or international. Another feature captured whether the flight occurred within the same geographical area, helping to identify flights that may share common air traffic control procedures or weather patterns.

To capture the flight direction, we created a feature that categorizes flights based on the regions of the departure and arrival airports. This classification enables the analysis of common flight routes and directional trends, offering insights into regional air traffic patterns. We also introduced a feature to identify intercontinental flights, which helps differentiate flights with varying characteristics due to longer distances, diverse regulations, and different airspace complexities.

Interaction Features We created interaction features to capture the combined effects of certain flight parameters. Specifically, we multiplied the average airspeed by the specific energy during different flight phases, resulting in new variables for the arrival, departure, and en-route phases. These features help us understand how an aircraft's speed influences its energy efficiency during different phases of flight.

Binning Continuous Variables We discretized certain continuous variables into bins to manage the effects of outliers and capture potential nonlinear relationships. We created temperature bins by categorizing the average temperature during the arrival phase into five equal-frequency categories. This allows the model to capture the impact of temperature variations on flight operations without being overly sensitive to extreme values. Similarly, we created humidity bins by categorizing the average humidity during the departure phase into five categories. Binning these variables facilitates the detection of patterns and thresholds in how temperature and humidity affect flight performance.

Ratio Features We engineered ratio features to analyze the relationship between vertical rate and airspeed during flight phases. The vertical rate indicates the rate of climb or descent, while the airspeed reflects the horizontal speed. The ratios provide insights into the aircraft's climb or descent efficiency relative to its speed during the arrival and departure phases.

2.3.2 Trajectory Data

The initial step involved calculating the distance from each data point of the trajectories dataset to the origin and destination airports. Using these distances, we segmented the trajectories into three phases: within 40 nautical miles of the origin airport, within 100 nautical miles of the destination airport, and the en route phase, which covers the segment beyond the radius of the origin and

destination airports. After the first data point en route or within the radius of the destination airport, all subsequent points are classified as en route or in the arrival phase, even if the flight returns to the previous airspace. For each phase, we computed the following variables:

- mean of vertical rate (ft/min), airspeed (m/s), ground speed (knots), temperature (Celsius), humidity, and altitude (ft);
- total wind distance (meters), i.e. the sum of the products of tailwind/headwind speed and time;
- flown distance (nautical miles);
- specific energy at the last data point ($V^2 * gh$, where V is the true airspeed (m/s), g is the gravity acceleration (m/s²) and h is the height (m)).

In addition to the variables computed for each flight phase, we also calculate other variables whose values may be impacted by the aircraft's takeoff weight.

- specific energy at the distance of 10 nautical miles flown;
- liftoff true airspeed and groundspeed;
- cruise altitude (ft);
- the 10 first airborne observations of true airspeed, temperature, height relative to origin airport, altitude, specific energy, and vertical rate;
- total time (min) and distance flown (nautical miles) in level flight after the takeoff and before the top of climb. The level segments were identified using a vertical speed limit of 300 feet per minute;
- total time and distance from take off to the top of climb.

2.4 Predictive Models

Given the popularity of ML models, there are many ML techniques available. To select which ones we would consider, we used our prior experience as ML practitioners and researchers together with knowledge from the literature [16, 17].

Despite Deep Learning dominating in perception tasks and in the big data regime, ML practitioners have noticed that gradient-boosting decision trees (GBDTs) often outperform neural networks on tabular data [17]. In fact, in Data Science competitions, the winning solutions usually employ GBDTs. In this regard, the libraries XGBoost [18], CatBoost [19], and LightGBM [20] are especially popular [17]. Therefore, inspired by our experience and the benchmark provided by [17], we decided to evaluate these three GBDT models and neural networks as our predictive models.

To ensure a fair comparison between the ML techniques, we standardized some parts of our training pipeline. The competition provided the challenge and submission datasets. The challenge dataset contains annotated labels, while the submissions dataset does not. We then separated the challenge dataset into training and test sets with an 80-20 split using the same random seed for every method. This intermediary test set was used to evaluate the performance locally.

Moreover, we executed hyperparameter optimization through the Optuna library [21] for each algorithm. Finally, before submitting the estimated ATOW in the submission dataset for evaluation by the challenge server, we retrained each model using the whole challenge dataset for the best possible performance.

To evaluate the performance of our predictive models, we used the root mean squared error (RMSE) as the primary metric, given its sensitivity to large errors, which is important in applications requiring high accuracy. RMSE measures the average squared differences between predicted and actual values, offering a clear view of the model's precision. Additionally, we implemented early stopping

to prevent overfitting and enhance model generalization. During training, the process halts if there is no improvement in the validation RMSE after 50 iterations, helping to avoid excessive fitting to the training data and ensuring robust performance on new data.

2.4.1 Artificial Neural Networks

This study employed two distinct artificial neural network architectures: a feedforward neural network implemented via FastAI's Tabular learner, and the Self-Attention and Intersample Attention Transformer (SAINT) model. These approaches represent different levels of complexity in tabular data processing, each with its own merits in terms of performance and computational requirements.

The FastAI Tabular learner implements a feedforward neural network, specifically optimized for tabular data. The architecture comprises an embedding layer for categorical variables, a preprocessing module for continuous variables, multiple fully connected layers, and an output layer. The embedding layer transforms categorical variables into dense vector representations, with the embedding dimensionality d for each variable determined by the formula: $d = min(600, round(1.6 * n\_categories^{0.56}))$, where *n_categories* is the number of unique categories in the variable. This adaptive dimensionality ensures appropriate representational capacity based on the cardinality of each categorical variable.

Continuous variables undergo standardization and, optionally, batch normalization to mitigate internal covariate shift. The network's core consists of a sequence of fully connected layers with Rectified Linear Unit (ReLU) activation functions. These layers can be customized in terms of width and depth, and may incorporate batch normalization and dropout regularization to enhance generalization. The output layer's architecture is task-dependent, utilizing a single unit with sigmoid activation for binary classification or multiple units with softmax activation for multi-class problems.

The Tabular learner incorporates several advanced optimization techniques. It employs a learning rate finder to determine optimal learning rate ranges, implements discriminative learning rates to allow differential learning speeds across network layers, and utilizes cyclical learning rate schedules to improve convergence characteristics and generalization performance.

In contrast, the SAINT model represents a more advanced approach, leveraging recent developments in transformer architectures. SAINT's primary innovation lies in its dual attention mechanism, which combines self-attention within samples and intersample attention across different samples in a mini-batch. This mechanism enables the model to capture complex intra- and inter-sample interactions, potentially leading to more nuanced feature representations.

SAINT initially projects both categorical and continuous variables into a unified embedding space, facilitating interactions between different feature types. The embedded representations are then processed through a stack of transformer encoder blocks. Each block comprises self-attention and intersample attention layers, followed by feed-forward networks. This architecture enables iterative refinement of representations, capturing increasingly complex patterns in the data.

A notable feature of SAINT is its ability to utilize contrastive pre-training. This unsupervised learning technique enhances the model's ability to learn robust representations by contrasting similar and dissimilar samples. The pre-training phase can be particularly advantageous in semi-supervised scenarios or when dealing with limited labeled data.

Empirically, SAINT has demonstrated superior performance on various tabular datasets, often surpassing both traditional gradient boosting methods and other neural network approaches. Its capacity to capture intricate relationships in the data renders it particularly suitable for complex tabular problems where simpler models may be inadequate. But when applied to this dataset, it's performance was the lowest.

The choice between the FastAI Tabular learner and SAINT depends on various factors including dataset complexity, available computational resources, model interpretability requirements, and specific performance criteria. The Tabular learner offers a more straightforward and computationally efficient approach, suitable for scenarios where interpretability is crucial or computational resources are constrained. Conversely, SAINT provides a more powerful and flexible approach, capable of modeling highly complex interactions in the data, albeit at the cost of increased computational complexity and potentially reduced interpretability.

Both models offer distinct advantages in the domain of tabular data analysis. The selection of an appropriate model should be based on a careful consideration of the specific requirements and constraints of the task at hand, potentially necessitating empirical evaluation of both approaches to determine the optimal solution.

2.4.2 CatBoost

According to Hancock and Khoshgoftaar [19], CatBoost is an open-source, gradient-boosted decision tree (GBDT) implementation optimized for effectively handling categorical data in supervised machine learning scenarios. Moreover, it introduces two main innovations: 1) Ordered Boosting, which enhances the handling of the order in which data is processed to avoid overfitting; 2) Ordered Target Statistics, a technique to process categorical variables by calculating statistics on the target variable. These features allow CatBoost to manage categorical data more efficiently than popular GBDT implementations like XGBoost or LightGBM, which often require extensive preprocessing to convert categorical data into numerical formats.

To optimize CatBoost's hyperparameters with Optuna [21], we used the Tree-structured Parzen Estimator (TSE) algorithm and the ranges presented in Table 1. Notice that the value for `reg_lambda` is searched in a logarithmic scale. Despite the `learning_rate` being an important hyperparameter, it is relatively easy to tune manually while having a too huge impact on the model's performance. In general, we noted that a smaller `learning_rate` resulted in a better model but took much longer to train. Hence, we decided to use a fixed `learning_rate` of 0.05 during the hyperparameter search for a maximum of 10,000 iterations. Then, we reduce it to 0.01 when training the final model for a maximum of 50,000 iterations. Furthermore, we employed early stopping with 50 of patience.

Finally, we extensively used the feature selection mechanism of CatBoost to determine which features should be eliminated from the model. Combining hyperparameter optimization and feature selection (both manually and through CatBoost's `select_features()`), we could improve the RMSE considerably.

Table 1. CatBoost hyperparameters and corresponding ranges used in Optuna.

Hyperparameter	Range	Type	Log Scale
`reg_lambda`	$[10^{-5}, 100]$	`float`	Yes
`random_strength`	$[10, 50]$	`float`	No
`depth`	$[1, 15]$	`int`	No
`min_data_in_leaf`	$[1, 30]$	`int`	No
`leaf_estimation_iterations`	$[1, 15]$	`int`	No

2.4.3 Light Gradient-Boosting Machine

LightGBM, or Light Gradient Boosting Machine, in Ke et al. [20], is an efficient gradient-boosting decision tree framework designed for distributed and efficient learning, particularly for large-scale and high-dimensional data.

According to Ke et al., LightGBM offers several advantages and innovative features compared to traditional gradient boosting methods: 1) Gradient-based One-Side Sampling improves the data sampling process. It ensures that the most informative instances with larger errors (and thus more significant gradients) are used for learning, enhancing the model's efficiency without compromising accuracy; 2) Exclusive Feature Bundling: reduces the dimensionality by bundling mutually exclusive features in high-dimensional data, many features are sparse, which means they contain primarily zeros; 3) Efficient Handling of Categorical Features: handles categorical features by value mapping, which is more efficient than the one-hot encoding used by many other algorithms; and 4) Leaf-wise Tree Growth: grows trees leaf-wise, it chooses the leaf that minimizes the loss for growth, allowing for better fitting models, often resulting in increased accuracy with fewer leaves.

2.4.4 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a highly efficient, scalable machine learning system for tree-based boosting, originally developed by Chen and Guestrin [18]. This algorithm builds upon traditional gradient boosting by introducing optimizations that make it particularly effective for large-scale and high-dimensional data. One of its core strengths lies in its ability to handle sparse data efficiently, allowing it to manage missing or zero-filled values without requiring imputation.

To accommodate diverse data distributions, XGBoost incorporates a weighted quantile sketch algorithm that approximates quantiles in cases where data points carry different weights or significance. This feature is especially beneficial for datasets with varying data densities and for models that need to prioritize certain data instances, such as in imbalanced classification tasks.

Additionally, XGBoost leverages parallelized tree construction and distributed computing capabilities, enabling it to handle massive datasets while maintaining low training times. It integrates regularization techniques, such as L1 and L2 regularization, which help reduce overfitting and improve the model's generalizability. These combined features make XGBoost one of the top choices for machine-learning competitions and real-world applications, often outperforming other algorithms in terms of accuracy and computational efficiency.

Key hyperparameters for XGBoost include `learning_rate`, `max_depth`, `min_child_weight`, `gamma`, `colsample_bytree`, `reg_alpha`, and `reg_lambda`. Each of these influences the model's complexity, regularization, and ability to generalize. Optimizing these parameters is important to achieve low error rates, and they were fine-tuned using a trial-based approach to minimize the RMSE.

The `learning_rate` controls the step size at each iteration, balancing convergence speed and the risk of missing the global minimum. The `max_depth` hyperparameter determines the maximum depth of each tree, impacting both model complexity and potential overfitting. The `min_child_weight` value is the minimum sum of instance weights needed to split a node, providing control for overfitting by limiting excessively deep trees. The `gamma` parameter introduces a minimum loss reduction for further partitioning, creating a threshold for additional split operations.

For feature sampling, `colsample_bytree` specifies the fraction of features to be randomly sampled at each tree, helping reduce the correlation between trees and increase model robustness. The `reg_alpha` and `reg_lambda` variables add L1 and L2 regularization terms, respectively, controlling the model's complexity and penalizing large coefficients to prevent overfitting.

Below is a table summarizing the values used during the hyperparameter optimization process:

2.4.5 Ensemble

According to [22], ensemble methods are learning algorithms that combine multiple models to make predictions on new data by aggregating their individual outputs. While ensembles are widely used

Table 2. XGBoost hyperparameters, corresponding ranges, and descriptions used in Optuna optimization.

Hyperparameter	Range	Description
`learning_rate`	[0.001, 0.1]	Step size per iteration
`max_depth`	[3, 15]	Max tree depth
`min_child_weight`	[1, 10]	Minimum instance weight sum for a node split
`gamma`	[0.001, 1.0]	Minimum loss reduction for splits
`colsample_bytree`	[0.4, 1.0]	Feature fraction for sampling per tree
`reg_alpha`	[0.0001, 1.0]	L1 regularization parameter
`reg_lambda`	[0.0001, 1.0]	L2 regularization parameter

in both classification and regression tasks, our focus is on regression, where the goal is to predict continuous values.

The theoretical justification for ensemble methods in regression can be understood through the bias-variance decomposition of the expected mean squared error (MSE). For any predictor $h(x)$, the expected MSE can be decomposed as:

$$\mathbb{E}[(y - h(x))^2] = \text{Bias}[h(x)]^2 + \text{Var}[h(x)] + \sigma^2,$$

where σ^2 is the irreducible error. When combining multiple models in an ensemble, if the individual models have similar bias but are diverse in their predictions (their errors are not perfectly correlated), the variance component of the error can be reduced while maintaining the bias. For m models with equal weights $\frac{1}{m}$, assuming equal variances σ_h^2 and average correlation ρ between model predictions, the variance of the ensemble becomes:

$$\text{Var}[f_{\text{ensemble}}] = \frac{\sigma_h^2}{m}(1 + (m-1)\rho).$$

This theoretical foundation motivates the use of weighted ensembles, where each model h in the hypothesis space $\mathcal{H}$ outputs a continuous prediction $h(x)$ for an input x. The ensemble combines these predictions through a weighted sum to produce the final prediction $f_{\text{ensemble}}(x)$:

$$f_{\text{ensemble}}(x) = \sum_{h \in \mathcal{H}} w_h h(x),$$

where w_h represents the weight assigned to model h, and $\sum_{h \in \mathcal{H}} w_h = 1$.

The optimal weights w_h can be viewed as the solution to a constrained optimization problem that minimizes the empirical risk on the validation set. In our approach, we use Quadratic Programming (QP) to find these weights by minimizing the MSE while ensuring the weights sum to one:

$$\begin{aligned} \min_{w} \quad & \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{h \in \mathcal{H}} w_h h(x_i)\right)^2 \\ \text{subject to} \quad & \sum_{h \in \mathcal{H}} w_h = 1, \\ & w_h \geq 0, \quad \forall h \in \mathcal{H}. \end{aligned}$$

This formulation has a parallel to the Bayesian model averaging approach, where instead of using posterior probabilities $P(h|S)$, we directly optimize the weights to minimize prediction error. The QP

optimization learns the relative importance of each model's predictions, potentially capturing their complementary strengths while accounting for correlations between their predictions.

In [23], Abdullahi et al. compare these algorithms according to the main features such as speed, scalability, and memory. These comparisons are in Table 3.

Table 3. Comparison of CatBoost, LightGBM, and XGBoost

Feature	CatBoost	LightGBM	XGBoost
Categorical Feature Handling	Best at handling categorical variables automatically.	Requires pre-processing for categorical variables.	Requires pre-processing for categorical variables.
Overfitting Robustness	Less prone to overfitting due to symmetrical trees.	Prone to overfitting if not careful with parameters.	Regularization helps prevent overfitting effectively.
Speed	Slower, especially on large datasets.	Fastest due to histogram-based algorithms.	Fast but generally not as fast as LightGBM.
Scalability	Good but less optimal for extensive datasets.	Excellent scalability to large datasets and high-dimensional data.	Good scalability, especially with system optimizations for parallel processing.
Memory Usage	Moderate	Low due to efficient data handling.	High, especially with large data sets.
Ease of Use	Easier with default settings handling categorical data well.	Requires careful data preparation and parameter tuning.	Requires significant parameter tuning and understanding of boosting.
Data Size Handling	Handles small datasets well.	Not as effective with small datasets.	Effective with both large and small datasets but requires tuning.

3. Discussions

This section presents a detailed discussion of the key findings from the study. We begin with an exploratory data analysis to reveal patterns and trends in the data. Next, we evaluate the predictive performance of the models, followed by an analysis of the importance of features to understand the factors influencing model predictions. Lastly, an ablation study is conducted to assess the contribution of different features and model components to overall performance.

3.1 Exploratory Data Analysis

We performed an exploratory analysis of the flight dataset to better understand its characteristics. Figure 4 shows the total number of flight observations for each month of 2022 and Figure 5 depicts the hourly distribution of actual departure times. The monthly distribution reveals that August 2022 accounted for the maximum number of flights, likely due to the increased demand volume typically seen for the summer vacation period in Europe. By contrast, the lowest number of operations was observed for February 2022. The hourly distribution indicates that traffic departing from European airports begins to rise in the early morning, reaching the peak at 9:00 UTC, when it starts to decrease until around 13:00 UTC. Traffic then builds up again in the afternoon, peaking at 16:00 UTC, before gradually declining until the end of the day.

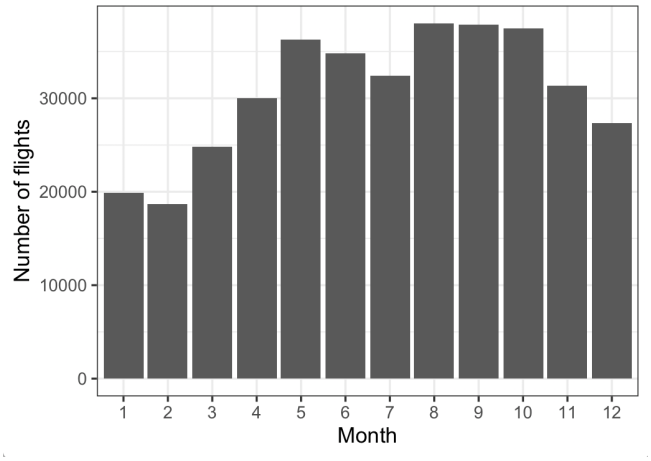

Figure 4. Total number of flight observations per month.

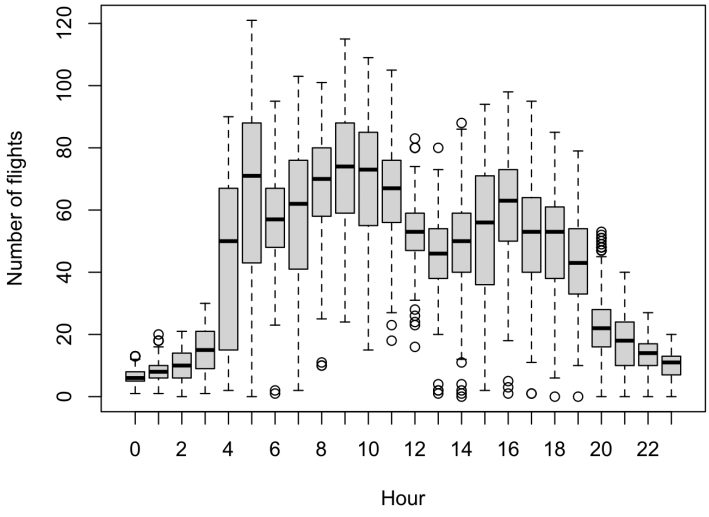

Figure 5. Hourly traffic distribution.

In terms of spatial coverage, the flight data contains observations for 4,228 origin-destination (OD) pairs, as shown in Figure 6. Most of the flights are within the European airspace, but the data also contains international flights between Europe and North, Central and South America, Africa, Middle East and Asia. In Europe, the maximum number of 2,157 flights was observed between London Heathrow Airport (EGLL) and Dublin International Airport (EIDW). Among the international routes, John F. Kennedy International Airport (KJFK) and London Heathrow Airport (EGLL) accounted for the maximum number of 763 flight observations.

The flights contained in the dataset were operated by 29 airlines using 30 different aircraft types, all pertaining to medium or heavy wake turbulence categories. Figures 7 and 8 shows the percentage

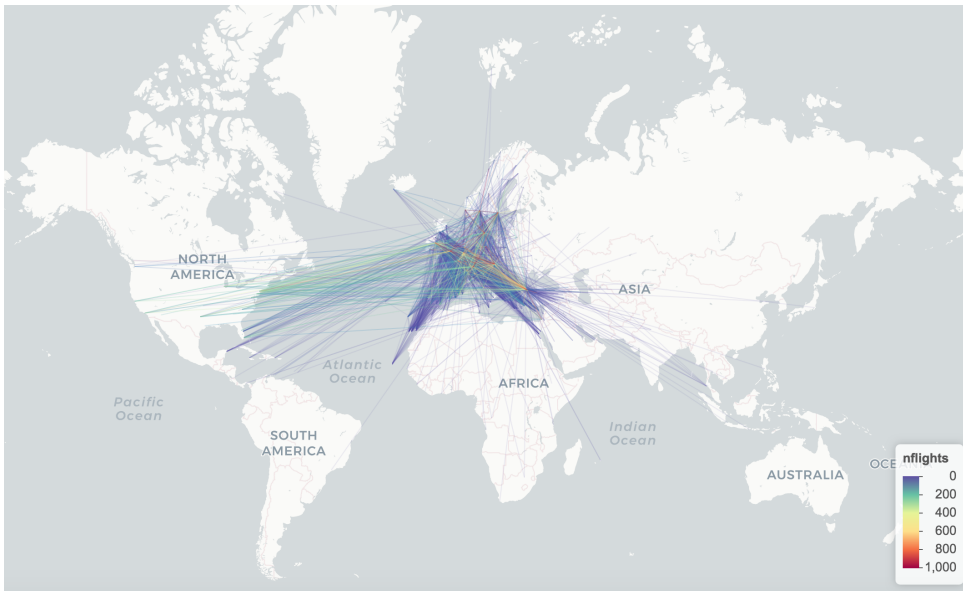

Figure 6. Origin-destination pairs and corresponding number of flight observations.

share of flight operations for each airline and equipment. More than 80% of flights were operated by 6 different airlines, with 2 airlines concentrating more than 40% of the operations. In terms of equipment, the distribution is slightly less uneven, with more than 80% of flights executed with 10 different types of aircraft. The A320 was the most used equipment, representing 21.6% of the observations.

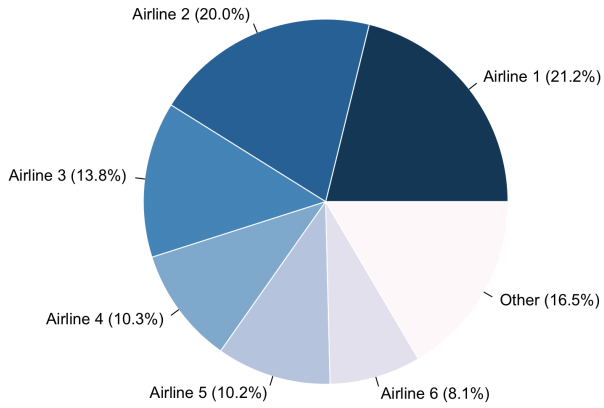

Figure 7. Share of flight operations by airline.

Figure 9 presents the distribution of ATOW for each aircraft type. As expected, the lowest ATOW was observed for the only medium turboprop aircraft (AT76). Among the medium jets, B752 showed the highest ATOW while CRJ9 showed the lowest. Heavy jets were inherently associated with the highest levels of ATOW. However, it is interesting to note a very high variability for some equipment, such as the B77W. Figure 10 reveals that a significant part of this variability is not explained by route length.

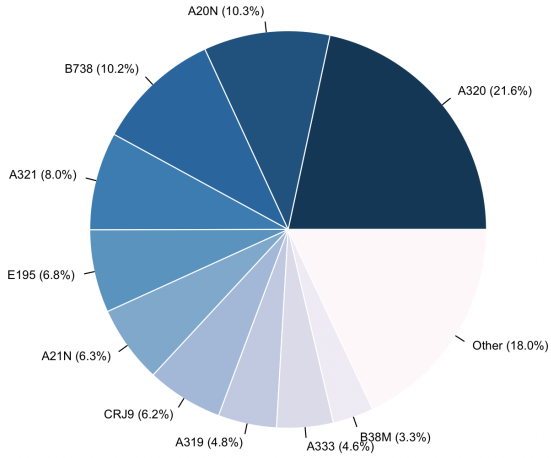

Figure 8. Share of flight operations by aircraft type.

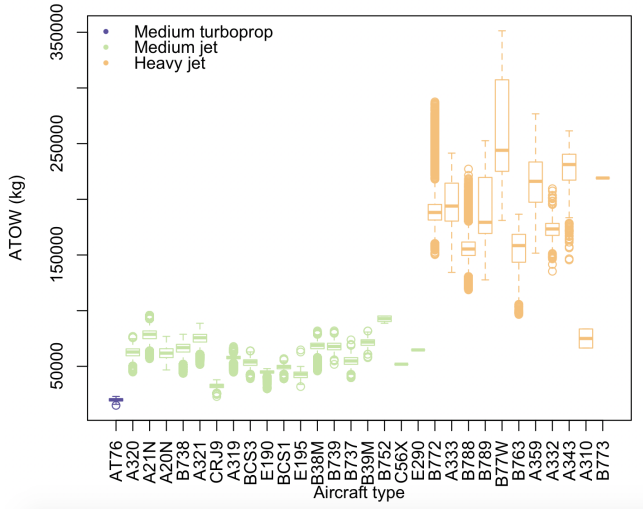

Figure 9. Boxplot of actual takeoff weight for each aircraft type.

3.2 Predictive Performance Analysis

This subsection presents the predictive performance analysis of five algorithms: XGBoost, CatBoost, LightGBM, ANN, and an ensemble combining all models. The analysis focuses on key metrics, including the best hyperparameters identified through optimization, the estimated RMSE based on the fabricated test dataset and the real RMSE provided by the challenge organizers.

The ensemble model combines predictions from the individual models, leveraging their complementary strengths to achieve improved predictive accuracy. The comparison results, shown in Table 4, demonstrate that the ensemble model achieved the best RMSE at 2200.00, highlighted in red as the lowest error among the models tested.

Each algorithm's estimated and real RMSE values align closely, suggesting that the models generalize well to new data. Among individual models, CatBoost achieved the lowest real RMSE at 2224.69,

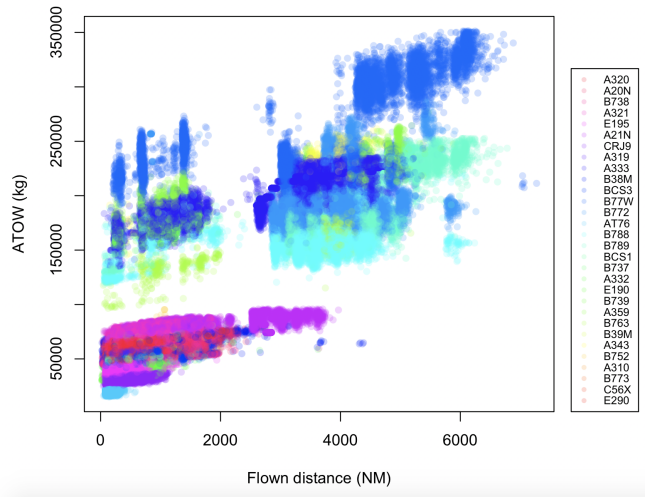

Figure 10. Scatterplot of the actual takeoff weight and distance flown for each aircraft type.

while LightGBM and XGBoost also demonstrated competitive performance. The ANN model, while performing well, showed slightly higher RMSE values compared to the gradient-boosted models, underscoring the effectiveness of Gradient-Boosted Decision Trees models on tabular data for this task.

3.3 Feature Importance Analysis

Figure 11 presents the top 20 feature importances obtained from the trained CatBoost's model. These importances were computed using the `PredictionValuesChange` method, which shows how much on average the prediction changes if the feature value changes. Notice that the aircraft features are the most important ones in general. Indeed, adding extra aircraft features from external datasets reduced the RMSE considerably when compared to using solely the features provided by the challenge.

3.4 Ablation Study

We present a comprehensive ablation study to quantify the impact of various feature groups and model components on the predictive performance of our Actual Takeoff Weight (ATW) estimation model. The study employs a systematic approach to feature and component removal, evaluating the consequent changes in RMSE.

The ablation process commenced with our optimal ensemble model as the baseline and also our best model when changing features. Features were categorized into five distinct groups: temporal, geographical, aircraft characteristics, operational, and meteorological. Each group was sequentially removed from the feature set, and the model was retrained and evaluated. Additionally, we assessed the impact of removing individual algorithms from the ensemble.

Table 5 presents the results of the ablation study, showcasing the RMSE values and percentage changes relative to the baseline model for each ablated configuration.

The ablation study reveals that aircraft characteristics ...

These findings shows the importance of comprehensive feature engineering in ATW prediction, particularly in relation to aircraft-specific and operational parameters. They also validate the efficacy

Table 4. Comparison of best hyperparameters and RMSE (estimated and real) for XGBoost, CatBoost, LightGBM, ANN, and the Ensemble models. The best RMSE values are highlighted in red.

Algorithm	Best Hyperparameters	Estimated RMSE	Real RMSE
ANN	`hidden_layers_size=[400, 300, 200], learning_rate=0.003`	3864.1	3864.2
CatBoost	`learning_rate=0.01, reg_lambda=0.05, random_strength=20.11, depth=9, min_data_in_leaf=11, leaf_estimation_iterations=12`	2215.53	2224.69
LightGBM	`learning_rate=0.01, reg_lambda=0.46, reg_alpha=0.17, min_child_weight=4, max_depth=13, colsample_bytree=0.60, objective=regression, random_state=42, n_estimators=50000, metric=rmse, n_jobs=-1, device=gpu, subsample=1.0`	2506.43	2519.17
XGBoost	`subsample=1.0, reg_lambda=0.46, reg_alpha=0.17, min_child_weight=4, max_depth=13, gamma=0.44, colsample_bytree=0.6`	2413.05	2403.07
Ensemble	`weighted average`	2200.00	2200.00

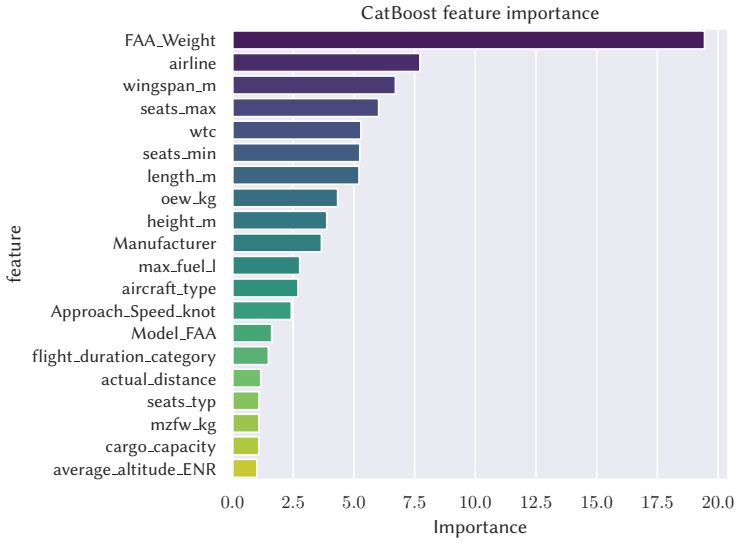

Figure 11. Top 20 feature importances for CatBoost.

of our ensemble approach.

4. Conclusion

Table 5. Ablation Study Results

Ablated Component	RMSE	Change in RMSE (%)
Baseline (Full Model)		
Temporal Features		
Geographical Features		
Aircraft Characteristics		
Operational Features		
Meteorological Features		
XGBoost Algorithm		
CatBoost Algorithm		
LightGBM Algorithm		
Neural Network		

Acknowledgement

Marcos Maximo is partially funded by CNPq – National Research Council of Brazil through the grant 307525/2022-8. Embraer funds Carolina Rutili through a scholarship associated with the Flight and Mobility Innovation Center (FLYMOV).

Author contributions

If the paper has more than one author, the CRediT section must be included. See example usage on https://casrai.org/credit/

- First Author: Conceptualization, Data Curation, Formal Analysis, Funding Acquisition, Investigation, Methodology, Project Administration, Resources, Software, Supervision, Validation, Visualization, Writing (Original Draft), Writing (Review and Editing)
- Second Author: Data curation, Writing- Original draft
- Third Author: Visualization, Investigation
- Marcos Maximo: Conceptualization, Data Curation, Methodology, Software, Supervision, Visualization, Writing (Original Draft), Writing (Review and Editing).

Funding statement

When applicable, please specify the funding information for this research.

Open data statement

DOI and short description to supplementary data.

Reproducibility statement

Information on how to reproduce this research, including access to 1) source code related to the research, 2) source code for the figures, and 3) source code/data for the tables when applicable.

References

[1] International Civil Aviation Organization. *Global Air Navigation Plan (Doc 9750, 6th Edition)*. ICAO, 2019.

[2] M. Murça and M. Oliveira. "A data-driven probabilistic trajectory model for predicting and simulating terminal airspace operations". In: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. 2020, pp. 1–7. DOI: 10.1109/DASC50938.2020.9256644.

[3] S. Mondoloni and N. Rozen. "Aircraft trajectory prediction and synchronization for air traffic management applications". In: *Progress in Aerospace Sciences* 119 (2020), p. 100640. DOI: https://doi.org/10.1016/j.paerosci.2020.100640.

[4] Y. Chati and H. Balakrishnan. "Statistical modeling of aircraft engine fuel flow rate". In: *30th Congress of the International Council of the Aeronautical Science*. 2016.

[5] C. Huang and X. Cheng. "Estimation of aircraft fuel consumption by modeling flight data from avionics systems". In: *Journal of Air Transport Management* 99 (2022), p. 102181. DOI: https://doi.org/10.1016/j.jairtraman.2022.102181.

[6] S. Badrinath, J. Abel, H. Balakrishnan, E. Joback, and T. Reynolds. "Spatial modeling of airport surface fuel burn for environmental impact analyses". In: *Journal of Air Transportation* 31.3 (2023), pp. 85–97. DOI: 10.2514/1.D0294.

[7] J. Sun, L. Basora, X. Olive, M. Strohmeier, M. Schäfer, I. Martinovic, and V. Lenders. "OpenSky Report 2022: Evaluating aviation emissions using crowdsourced open flight data". In: *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*. 2022, pp. 1–8. DOI: 10.1109/DASC55683.2022.9925852.

[8] R. Teoh, Z. Engberg, M. Shapiro, L. Dray, and M. Stettler. "The high-resolution Global Aviation emissions Inventory based on ADS-B (GAIA) for 2019–2021". In: *Atmospheric Chemistry and Physics* 24.1 (2024), pp. 725–744. DOI: 10.5194/acp-24-725-2024.

[9] R. Coppenbarger. "En route climb trajectory prediction enhancement using airline flight-planning information". In: *Guidance, Navigation, and Control Conference and Exhibit*. 2019. DOI: 10.2514/6.1999-4147.

[10] R. Alligier, D. Gianazza, and N. Durand. "Learning the aircraft mass and thrust to improve the ground-based trajectory prediction of climbing flights". In: *Transportation Research Part C: Emerging Technologies* 36 (2013), pp. 45–60. DOI: https://doi.org/10.1016/j.trc.2013.08.006.

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

[12] R. Alligier. "Predictive Distribution of Mass and Speed Profile to Improve Aircraft Climb Prediction". In: *Journal of Air Transportation* 28.3 (2020), pp. 114–123. DOI: 10.2514/1.D0181.

[13] J. Sun, J. Ellerbroek, and J. Hoekstra. "Modeling and Inferring Aircraft Takeoff Mass from Runway ADS-B Data". In: *7th International Conference on Research in Air Transportation*. 2016.

[14] J. Sun, J. Ellerbroek, and J. Hoekstra. "Aircraft initial mass estimation using Bayesian inference method". In: *Transportation Research Part C: Emerging Technologies* 90 (2018), pp. 59–73. DOI: https://doi.org/10.1016/j.trc.2018.02.022.

[15] Eurocontrol. *Performance Review Commission (PRC) Data Challenge*. [Online; accessed September 2024]. URL: https://ansperformance.eu/study/data-challenge/.

[16] Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. "Why do tree-based models still outperform deep learning on typical tabular data?" In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 507–520. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/0378c7692da36807bdec87ab043cdadc-Paper-Datasets_and_Benchmarks.pdf.

[17] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. "Why do tree-based models still outperform deep learning on typical tabular data?" In: *Proceedings of the 36th International Con-*

ference on Neural Information Processing Systems. NIPS '22. New Orleans, LA, USA: Curran Associates Inc., 2024. ISBN: 9781713871088.

[18] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145/2939672.2939785.

[19] John T Hancock and Taghi M Khoshgoftaar. "CatBoost for big data: an interdisciplinary review". In: *Journal of big data* 7.1 (2020), p. 94.

[20] 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: *Advances in neural information processing systems*. 2017, pp. 3146–3154.

[21] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 2623–2631. ISBN: 9781450362016. DOI: 10.1145/3292500.3330701. URL: https://doi.org/10.1145/3292500.3330701.

[22] Thomas G Dietterich. "Ensemble methods in machine learning". In: *International workshop on multiple classifier systems*. Springer. 2000, pp. 1–15.

[23] Abdullahi A Ibrahim, Raheem L Ridwan, Muhammed M Muhammed, Rabiat O Abdulaziz, and Ganiyu A Saheed. "Comparison of the CatBoost classifier with other machine learning methods". In: *International Journal of Advanced Computer Science and Applications* 11.11 (2020).