Journal of Open Aviation Science (2025), Vol.**3**
doi:10.59490/joas.2025.7963

JOAS

ARTICLE

Open Machine Learning Models for Actual Takeoff Weight Prediction

Mayara C. R. Murça, *,[1] Marcos R. O. A. Maximo ,[2] Joao P. A. Dantas ,[3] João B. T. Szenczuk ,[1] Carolina R. Lima ,[2] Lucas O. Carvalho ,[1] and Gabriel A. Melo [2]

[1]Civil Engineering Division, Aeronautics Institute of Technology, São José dos Campos, Brazil
[2]Autonomous Computational Systems Lab (LAB-SCA), Computer Science Division, Aeronautics Institute of Technology, São José dos Campos, Brazil
[3]Decision Support Systems Subdivision, Institute for Advanced Studies, São José dos Campos, Brazil
*Corresponding author: mayara@ita.br

(Received: 20 Dec 2024; Revised: 24 Mar 2025; Accepted: 24 Mar 2025; Published: 8 Apr 2025)
(Editor: Xavier Olive; Reviewers: Richard Alligier, Ryota Mori, and Junzi Sun)

Abstract
Aircraft weight is a key input in flight trajectory prediction and environmental impact assessment tools. However, the lack of openly available data regarding the actual aircraft weight throughout the flight requires the development of mass estimation approaches to be incorporated into these tools. This study uses large-scale open aviation data made available by Eurocontrol's Performance Review Commission to develop an open-source machine learning model to predict commercial flights' actual takeoff weight. The data combines detailed flight, trajectory, and meteorological information for 369,013 flights that transited through the European airspace in 2022. Several operational features are created to represent each flight's horizontal and vertical profiles accurately. For model learning, we employ *CatBoost*, *LightGBM*, *XGBoost*, artificial neural networks, and an ensemble of these models, which were selected for their robust performance in structured data analysis and potential for high predictive accuracy. The models are evaluated based on their efficiency, accuracy, and applicability to real-world data. The best-performing model is found to predict the aircraft takeoff weights with a mean percentage of error of 1.73%.

Keywords: Air traffic management; aircraft mass; predictive modeling; machine learning

1. Introduction

Advances in Air Traffic Management (ATM) are important to achieve 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 environmental 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 proposes an open-source machine learning model to predict a flight's actual takeoff weight (ATOW) based on large-scale open aviation data. The machine learning methods applied in this study are *LightGBM*, *CatBoost*, *XGBoost*, artificial neural networks, and an ensemble model combining these algorithms, chosen for their effectiveness in handling structured data and potential for high predictive accuracy. The choice of these methods reflects the popularity of Gradient-Boosting Decision Trees (GBDT) models for tabular data and the versatility of neural networks for complex data patterns. The main motivation for this work was 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 predictive models. In Section 3, we discuss the results, comparing and analyzing different machine learning 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. We describe the datasets and the steps taken to prepare and enhance them, focusing on how meaningful features were extracted for model learning. The trajectory data processing was developed in R, while the feature engineering and the machine learning models were implemented in Python. Our code is released as open source [1].

2.1 Data Description

This subsection details the open-source datasets used in the study, including flight and trajectory data and 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.

[1]https://github.com/PRC-Data-Challenge-2024/team_tiny_rainbow

2.1.1 Flight and Trajectory Data

Through the PRC Data Challenge, Eurocontrol provides the primary data used in this work to predict a flight's actual takeoff weight.

The challenge data made available for model learning is composed of two datasets. The first contains flight data for 369,013 flights that departed or arrived at a European airport in 2022. It provides flight identification, origin, destination airports, aircraft type and wake turbulence category, airline, and operational parameters such as 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 due to limited ADS-B coverage in some parts of the world and lower altitudes.

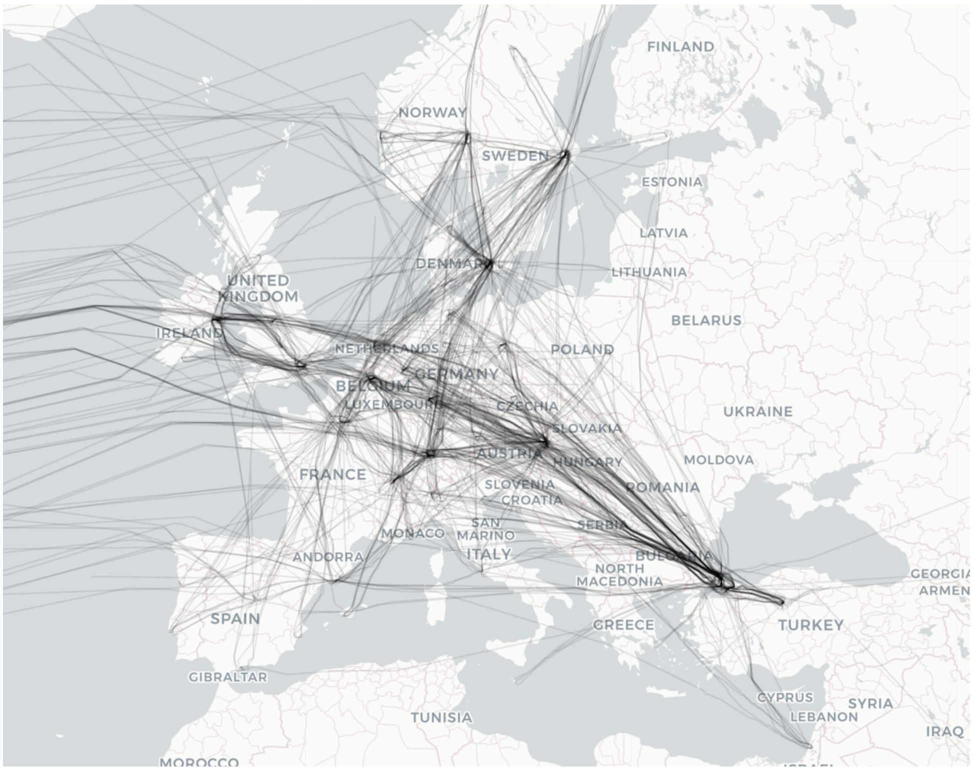

Figure 1. Horizontal profile of trajectories flown in the European airspace on April 2, 2022.

The Data Challenge provided a final submission dataset with flight and trajectory data for an additional set of 158,149 flights to rank the participating teams. We also used this dataset to evaluate the final performance of the models learned based on the predictive performance metrics computed by the Data Challenge organizers.

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[2] and the Global Airport Database[3]. While these sources provided extensive coverage, information about 15 airports in both the challenge and submission sets was still lacking, which we manually extract from Google Maps and added to the dataset to ensure completeness. Additionally, we incorporated an open-source dataset from the Federal Aviation Administration (FAA)[4], which contains detailed aircraft characteristics. This dataset, updated in September 2023, provides aircraft performance parameters and operational characteristics for our analysis. Furthermore, we created a supplementary database to capture additional aircraft features not present in the FAA dataset. This extra 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. 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 ensured consistency in each flight record's temporal and spatial attributes, enabling accurate tracking of aircraft positions and speeds over time. Also, to analyze the various phases of flight, each trajectory was segmented into three key stages: terminal area departure, en route, and terminal area arrival. We follow the flight phase segmentation currently used for ATM performance analysis, modeling the departure/arrival terminal areas with cylindrical volumes with centers at the origin/destination airports and with a radius of 40 NM/100 NM, respectively.

Thus, the departure phase is defined as the segment of the flight trajectory from takeoff until the aircraft exits a 40 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 cylindrical volumes. This approach ensures that each flight phase is accurately identified, facilitating subsequent analyses of the aircraft behavior in different flight stages. As an example, Figure 2 shows the trajectory of a flight from Lyon–Saint Exupéry Airport (LFLL) to Brussels Airport (EBBR), segmented by flight phase.

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.

We engineered several new features to capture horizontal and vertical profile characteristics of flight trajectories, temporal dynamics, flight efficiency, geographical context, and the relationship between

[2]OpenFlights Airports Database: https://raw.githubusercontent.com/jpatokal/openflights/master/data/airports.dat
[3]Global Airport Database: https://www.partow.net/miscellaneous/airportdatabase/
[4]FAA Aircraft Characteristics Database: https://www.faa.gov/airports/engineering/aircraft_char_database/data

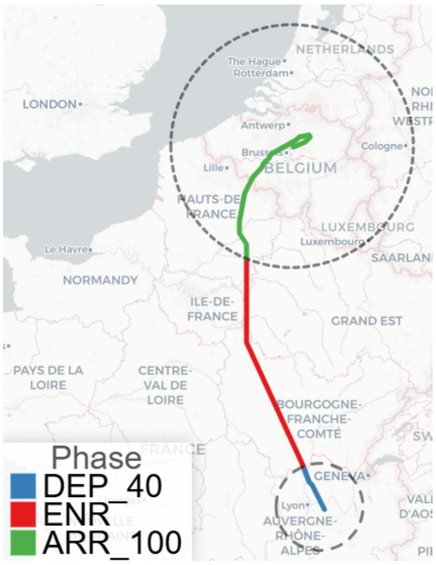

Figure 2. Segmentation of flight phases for a flight from Lyon–Saint Exupéry Airport (LFLL) to Brussels Airport (EBBR).

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.

For the interested reader, the complete feature engineering code is available in a notebook that includes detailed comments, explanations, and intermediate printouts to help readers understand the processing and transformation applied to each feature used in the analysis. The link to the repository is provided at the end of this work. Due to the large number of variables used, we did not provide a complete list in the manuscript, as to avoid taking up too much space, but we describe the process of obtaining these variables throughout this section.

2.3.1 Flight Trajectory Features

Following the identification of flight phases, we created several features to describe the actual trajectory profiles and the meteorological conditions during the operation. For each flight phase (departure, en route, and arrival), we computed the following aggregate features:

- mean of vertical rate (ft/min), airspeed (m/s), squared airspeed, ground speed (kt), temperature (K), humidity (g/kg), and altitude (ft);
- maximum altitude (ft), defined as the 99^{th} percentile of altitude values to mitigate the impact of noisy observations;
- total wind distance (m), i.e., the sum of the products of tailwind/headwind speed and time;
- flown distance (NM);
- specific energy at the last data point ($V^2 + gh$), where V is the airspeed (m/s), g is the gravity acceleration (m/s^2) and h is the height (m)).

Specifically for the departure phase, for flights with the first observation within 2 nautical miles of the origin airport, we enriched the dataset with the time series of airspeed, squared airspeed, temperature, height relative to the origin airport, specific energy, and vertical rate for the first ten observations after takeoff. Additionally, we included information regarding the efficiency of the vertical profile from takeoff until the top-of-climb. We computed the total time (min) and distance flown (NM) in level flight from takeoff until the top-of-climb, as level-off segments tend to generate higher fuel burn. Particularly in the departure phase, these level segments might be more probably associated with structural inefficiencies rather than operational inefficiencies, potentially being accounted for during fuel planning. Moreover, step climbs may correlate with aircraft weight. As fuel is consumed and aircraft mass diminishes, the aerodynamic efficiency at higher altitudes improves, necessitating incremental climbs to optimize fuel consumption. Furthermore, heavier aircraft may exhibit reduced initial climb performance, rendering step climbs a strategic method to achieve optimal flight levels. To compute the vertical efficiency during the climb, we follow Eurocontrol's methodological approach [16]. For the trajectory portion within 200 NM from the origin airport and 3,000 ft above ground level, we identify the highest altitude reached as the Top-of-Climb (TOC). An exclusion box is defined around the portion of the vertical profile for which the altitude is greater than or equal to 90% of the TOC. Climb segments with a rate of climb smaller than or equal to 300 ft/min are then identified as level-off segments, provided they occur outside the exclusion box if lasting more than 5 minutes. The distance and time flown in level segments are calculated and summed to produce the features. The approach is illustrated in Figure 3.

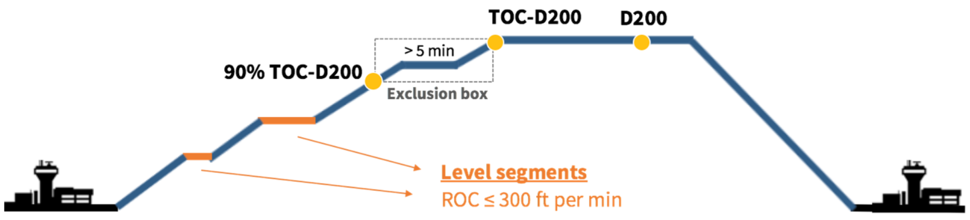

Figure 3. Illustration of the identification of level segments during climb for vertical efficiency quantification.

Finally, we created additional features to describe operational features that are expected to be correlated with the aircraft's takeoff weight based on prior literature. These include the specific energy at the distance of 10 NM flown [17] and the liftoff airspeed and groundspeed [14].

2.3.2 Temporal Features

From the departure time, we extracted attributes such as the hour 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 weekends or during rush hours (defined as 7–9 AM or 4–6 PM based on our exploratory data analysis). These features help assess the impact of higher traffic volumes on flight performance.

2.3.3 Duration-Based Features

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 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 may help segment flights for analysis and modeling purposes. Grouping durations into categories could allow the model to recognize patterns in different types of flights, making it easier to capture potential operational differences, such as fuel usage and air traffic procedures.

2.3.4 Geographical Features

Using the Haversine formula to incorporate geographical context and spatial relationships, we calculated the distance between the departure and arrival airports. It 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.

2.3.5 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, generating new variables for the arrival, departure, and en route phases. This interaction helps quantify how velocity contributes to the aircraft's total energy state, which is a key factor in assessing maneuverability and efficiency. By incorporating these features, we aim to analyze how variations in speed influence the aircraft's energy availability and operational performance across different stages of flight.

2.3.6 Aircraft Characteristics Features

In addition to the flight trajectory features, we incorporated several aircraft-specific characteristics to enhance our predictive models. These features capture the physical and operational attributes of each aircraft type that influence takeoff weight. We included the Maximum Takeoff Weight (MTOW), which represents the maximum weight at which the aircraft is certified for takeoff, serving as an upper limit for our predictions. The Operating Empty Weight (OEW) was also incorporated, representing the basic weight of the aircraft including crew but excluding usable fuel and payload. Maximum payload capacity was considered to account for the weight limitations of passengers, cargo, and baggage. Fuel capacity data provided information on the maximum amount of fuel each aircraft can carry, directly impacting potential takeoff weight. We also utilized engine specifications, including type and count, which affect fuel consumption and performance characteristics. Aircraft age categorization (new, mid-life, or older generation) was included to reflect technological advancements in fuel efficiency. Seating configuration data provided insight into standard passenger capacity, while physical dimensions such as wing span and area were incorporated to account for aerodynamic performance factors. Finally, we included range capability information to represent the maximum distance each aircraft can fly without refueling under standard conditions. These aircraft-specific features provide the model with essential context about the physical constraints and operational capabilities of each aircraft type, which are fundamental determinants of takeoff weight.

2.3.7 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.

2.3.8 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.

2.4 Predictive Models

Given the popularity of machine learning models, there are many techniques available. To select which ones we would consider, we used our prior experience as machine learning practitioners and researchers together with knowledge from the literature [18].

Despite Deep Learning dominating perception tasks and big data regimes, machine learning practitioners have noticed that Gradient-Boosted Decision Trees (GBDT) often outperform neural networks on tabular data [18]. In fact, the winning solutions in Data Science competitions usually employ GBDTs. In this regard, the methods XGBoost [19], CatBoost [20], and LightGBM [21] are especially popular [18]. In [22], Abdullahi et al. compare these algorithms according to the main features such as speed, scalability, and memory. These comparisons are in Table 1. Therefore, based on our experience and the benchmark provided by [18], we decided to evaluate these three GBDT methods and neural networks for our predictive models.

We standardized parts of our training pipeline to ensure a fair comparison between the machine learning methods. We separated the dataset into training and test sets with an 80-20 split using the same random seed for every technique. Since the trajectory data was incomplete for all flights, some of the trajectory features presented missing values. Therefore, we tried different methods of data imputation. Experimentally, we found that using the median of the valid elements of the respective column worked the best, outperforming more advanced techniques, such as linear regression and k-NN. This behavior was consistent across the different learning algorithms.

Furthermore, we executed hyperparameter optimization through the Optuna library [23] for each

Table 1. Comparison of features 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.

method. We also implemented early stopping to prevent overfitting and enhance model generalization. During training, the process halts if there is no improvement in the validation performance metric after 50 iterations, helping to avoid excessive fitting to the training data and ensuring robust performance on new data. Before submitting the estimated ATOW in the submission dataset for evaluation by the Data Challenge server, we retrained each model using the whole challenge dataset for the best possible performance. Specific details of each learning algorithm are presented in the following subsections.

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 [24] and the Self-Attention and Intersample Attention Transformer (SAINT) model [25]. 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 multilayer perceptron (MLP) optimized explicitly 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

$$d = \min(600, \mathrm{round}(1.6 * \mathrm{n\_categories}^{0.56})), \tag{1}$$

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 shifts. The network's core consists of a fully connected layer sequence with Rectified Linear Unit (ReLU) activation functions. These layers can be customized regarding 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, taking advantage of 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 intersample interactions, potentially leading to more nuanced feature representations.

SAINT initially projects 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, where each block comprises self-attention and intersample attention layers, followed by feed-forward networks, which allow an 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 traditional gradient-boosting methods and other neural network approaches as its capacity to capture more relationships in the data makes it particularly suitable for complex tabular problems where simpler models may be inadequate. However, when applied to this dataset, its performance was the lowest, which could be attributed to the regression nature of the problem and the fact that most input variables were continuous and not categorical, in contrast with most tabular datasets.

The models were trained for up to 50 epochs, with early stopping using a maximum learning rate of 0.005 that decreased according to a cosine annealing schedule. This learning rate was found following the super-convergence heuristic, which is the learning rate one order of magnitude lower than the one in which the loss would diverge. We used the Adam (Adaptive Gradient Estimation) optimizer to update the network's parameters given the gradients of the loss computed by backpropagation. While we did not automate the search of the network's hyperparameters using Optuna, we performed a simple sweep over various network widths ranging from 32 up to 256 neurons per layer.

Both neural network architectures were trained with carefully selected hyperparameters, detailed in Table 2. For the FastAI Tabular learner, we used a batch size of 128, which balanced computational efficiency with model convergence quality. The network was trained using the Adam optimizer with a maximum learning rate of 0.005 controlled by a cosine annealing schedule. We employed early stopping with a patience of 50 epochs to prevent overfitting. The model's performance was evaluated using the root mean squared error metric, scaled appropriately to the range of the target variable.

For the SAINT model, we explored multiple hyperparameter configurations through a systematic manual search. The optimal configuration included an embedding size of 32, transformer depth of 6 with 8 attention heads, attention dropout of 0.1, and feed-forward dropout of 0.1. The model was trained with a batch size of 2048 using the AdamW optimizer with a learning rate of 0.002 for up to 50 epochs. We incorporated both contrastive and denoising pre-training tasks for 15 epochs prior to supervised training, which helped the model learn robust representations from the tabular data. Model selection was performed using a dedicated validation set (5% of the training data), and we saved the model with the lowest validation error for final evaluation.

Table 2. Neural network hyperparameters

Hyperparameter	FastAI Tabular Learner	SAINT Model
Batch size	128	2048
Learning rate	0.005	0.002
Optimizer	Adam	AdamW
Embedding size	Adaptive	32
Hidden layers	[400, 300, 200]	-
Transformer depth	-	6
Attention heads	-	8
Dropout	0.1	0.1 (attention), 0.1 (feed-forward)
Pre-training	No	Yes (contrastive and denoising)
Early stopping patience	50	50

The selection of these values was based on a combination of literature recommendations, empirical observations during preliminary experiments, and resource constraints. While we did not employ automated hyperparameter optimization for the neural networks due to computational limitations, our manual tuning process was guided by established best practices for tabular data modeling.

2.4.2 CatBoost

According to Hancock and Khoshgoftaar [20], CatBoost is an open-source GBDT implementation optimized to effectively handle 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 [23], we used the Tree-structured Parzen Estimator (TSE) algorithm and the ranges presented in Table 3. 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 significantly impacting the model's performance. We noted that a smaller `learning_rate` generally 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 100,000 iterations.

We highlight that the final training took around 4 hours on a high performance computer with an Intel Core i9-7900X @ 3.30 GHz processor and a NVIDIA GTX 1080 GPU card. Furthermore, we employed early stopping with 50 of patience, both during the hyperparameter search and the final training.

The tuned hyperparameters can be briefly described as follows. The `learning_rate` controls the step size of an update during training, with lower values enhancing generalization but increasing training time. `reg_lambda` is the coefficient of the L2 regularization in the cost function, used to penalize large model weights. `random_strength` influences the amount of randomness used to score splits, which helps enhancing robustness. `depth` defines the depth of the decision trees in order to control the trade-off between model complexity and overfitting. `min_data_in_leaf` sets the minimum number of samples required in a leaf, so to avoid learning from noise. The hyperparameter `leaf_estimation_iterations` sets how many iterations are used to estimate leaf values. For more information, we refer the interested reader to the CatBoost's documentation.

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 improved model performance considerably.

Table 3. 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. [21], is an efficient GBDT framework designed for distributed and efficient learning, particularly for large-scale and high-dimensional data.

According to Ke et al. [21], 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 and chooses the leaf that minimizes the loss for growth, allowing for better fitting models, often resulting in increased accuracy with fewer leaves.

Likewise, we employed Optuna [23] to tune the best hyperparameters of LightGBM, using the ranges presented in Table 4. The training of the LightGBM model with Optuna involved setting up an early stopping mechanism and logging the evaluation results during each iteration.

The hyperparameters shown in Table 4 can be briefly described as: the `learning_rate` defines the step size of the model adjustment at each step; `max_depth` limits the depth of trees to prevent overfitting; `min_child_weight` sets the minimum sum of instance weights in a leaf; `colsample_bytree` defines the fraction of features randomly chosen for each tree, improving diversity and reducing overfitting; `reg_alpha` applies L1 regularization while `reg_lambda` applies L2 regularization; and `n_estimators` specifies the number of boosting iterations. For more information, we refer the interested reader to the LightGBM's documentation.

Table 4. LightGBM hyperparameters and corresponding ranges used in Optuna.

Hyperparameter	Range	Type	Log Scale
learning_rate	[1e-3, 0.1]	float	Yes
max_depth	[3, 15]	int	No
min_child_weight	[1, 10]	int	No
colsample_bytree	[0.4, 1.0]	float	No
reg_alpha	[1e-4, 1.0]	float	Yes
reg_lambda	[1e-4, 1.0]	float	Yes
n_estimators	1,000,000	int	No

2.4.4 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a highly efficient, scalable machine learning system for tree-based boosting, developed initially by Chen and Guestrin [19]. 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 is its ability to handle sparse data efficiently, allowing it to manage missing or zero-filled values without imputation.

XGBoost incorporates a weighted quantile sketch algorithm to accommodate diverse data distributions. This algorithm approximates quantiles 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 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 generalization ability.

The `learning_rate` controls the step size at each iteration. 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, controlling overfitting by limiting intense 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. Table 5 summarizes the values used during the hyperparameter optimization process.

2.4.5 Ensemble

According to [26], ensemble methods are learning algorithms that combine multiple models to make predictions by aggregating their outputs. While ensembles are widely used in both classification and regression tasks, our focus is on regression for predicting continuous values.

Table 5. XGBoost hyperparameters, corresponding ranges, and descriptions used in Optuna optimization.

Hyperparameter	Range	Type	Log Scale
`max_depth`	[3, 15]	int	No
`min_child_weight`	[1, 10]	int	No
`gamma`	[0.001, 1.0]	float	Yes
`colsample_bytree`	[0.4, 1.0]	float	No
`reg_alpha`	[0.0001, 1.0]	float	Yes
`reg_lambda`	[0.0001, 1.0]	float	Yes

Ensemble methods can be broadly categorized into non-adaptive approaches (such as bagging, where models are combined with equal weights) and adaptive approaches (such as stacking or Super Learner [27], where model weights are learned from data). In this work, we implement an adaptive ensemble approach similar to stacking [28], which optimizes the combination weights to minimize prediction error.

For our ensemble, 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:

$$f_{\text{ensemble}}(x) = \sum_{h \in \mathcal{H}} w_h h(x), \tag{2}$$

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

To determine the optimal weights, we use a holdout validation set (20% of the original training data) that was not used in training the individual models. The weights are learned by solving a constrained optimization problem that minimizes the empirical risk on this validation set using Quadratic Programming (QP):

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

This optimization approach has several advantages over simple averaging. First, it can give higher weight to models that perform better than others on the training set. Second, it can deal with correlated predictions between models by adjusting their relative contributions. Third, it can completely exclude models that do not add predictive value beyond what is already captured by other models in the ensemble.

The theoretical benefits of such weighted ensembles have been well-established in the literature. Breiman [28] showed that stacked regression can achieve lower prediction error than any of the individual models, while van der Laan et al. [27] demonstrated that the Super Learner approach is asymptotically optimal among the class of all weighted combinations of the candidate learners.

To find the optimal weights for our ensemble model, we employed a two-stage process. First, we split our available data into two portions: 80% for training the individual models (CatBoost, LightGBM, XGBoost, and neural networks), and a held-out 20% validation set specifically for computing the optimal ensemble weights using the quadratic programming approach described above. This separation ensures that the weights are learned on data not seen during individual model training, helping to avoid potential overfitting. After the weights are learned, we retrain each individual model on the full training dataset before applying the ensemble to make predictions on new data. For the final evaluation on the submission dataset, all models were retrained using the complete challenge dataset with the pre-computed optimal weights applied to their predictions.

2.5 Predictive Performance Evaluation

To evaluate the performance of our predictive models, we used the Root Mean Squared Error (RMSE) as the primary metric, given its suitability for regression problems and sensitivity to large errors, which is essential in applications requiring high predictive performance. RMSE measures the average squared differences between predicted and actual values:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{3}$$

where y_i and $\hat{y}_i$ are the i-th actual and predicted values, respectively. Additionally, we computed the Mean Absolute Percentage Error (MAPE) to obtain a more intuitive estimate of the prediction error as a percentage of the actual output. Mathematically, MAPE is defined as

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100. \tag{4}$$

3. Discussions

This section presents a detailed discussion of the study's key findings. We begin with an exploratory data analysis to identify 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 model components to overall performance.

3.1 Exploratory Data Analysis

We performed an exploratory analysis of the flight dataset to understand its characteristics better. 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 European summer vacation period. By contrast, the lowest number of operations was observed for February 2022. The hourly distribution indicates that traffic departing from European airports rises in the early morning, reaching the peak at 9:00 UTC, when it decreases until around 13:00 UTC. Traffic builds up again in the afternoon, peaking at 16:00 UTC, before gradually declining until the end of the day.

Regarding 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, the Middle East, and Asia. The maximum number of 2,157 European 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 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 show each airline and aircraft

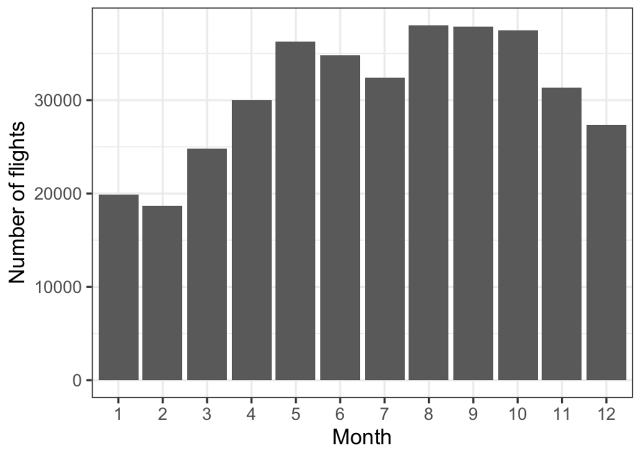

Figure 4. Total number of flight observations per month.

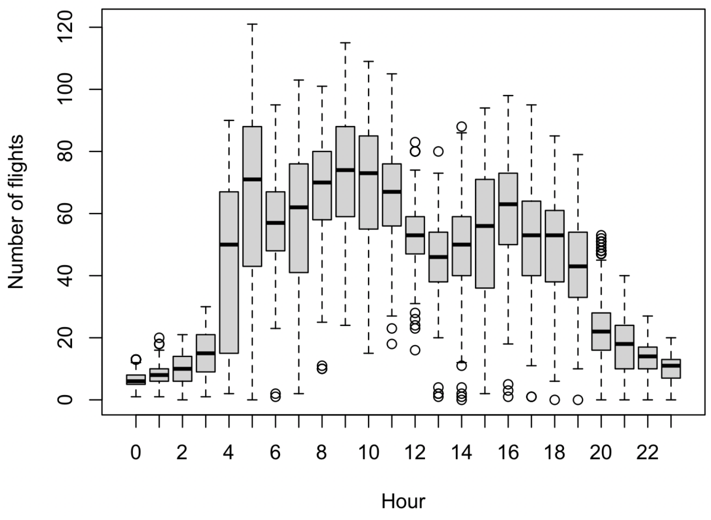

Figure 5. Hourly traffic distribution.

type's percentage share of flight operations. More than 80% of flights were operated by six different airlines, with two airlines concentrating more than 40% of the operations. In terms of aircraft, the distribution is slightly less uneven, with more than 80% of the flights executed with ten different types. The A320 was the most used aircraft type, representing 21.6% of the observations.

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, we observed a very high variability for some models, such as the B77W.

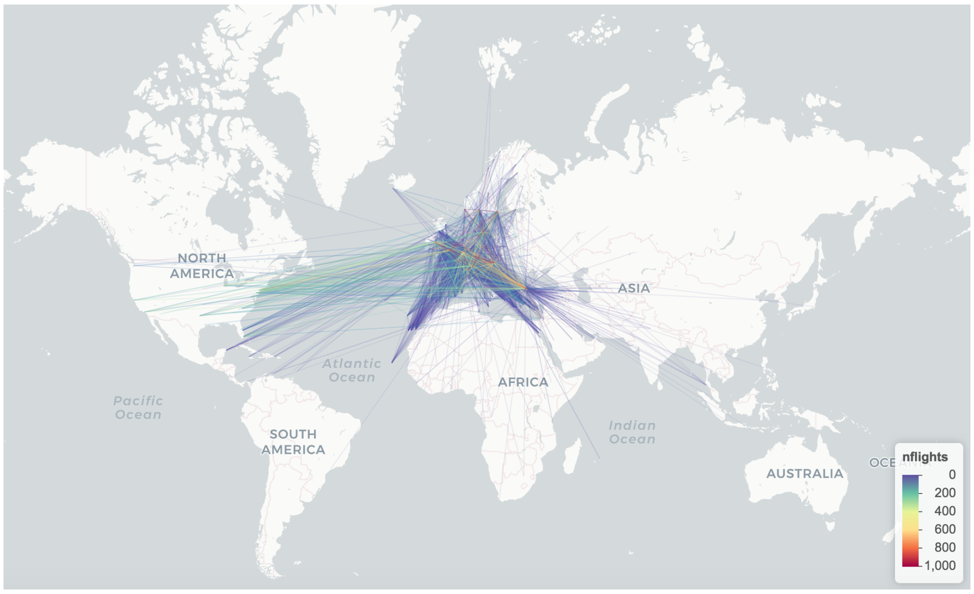

Figure 6. Origin-destination pairs and corresponding number of flight observations.

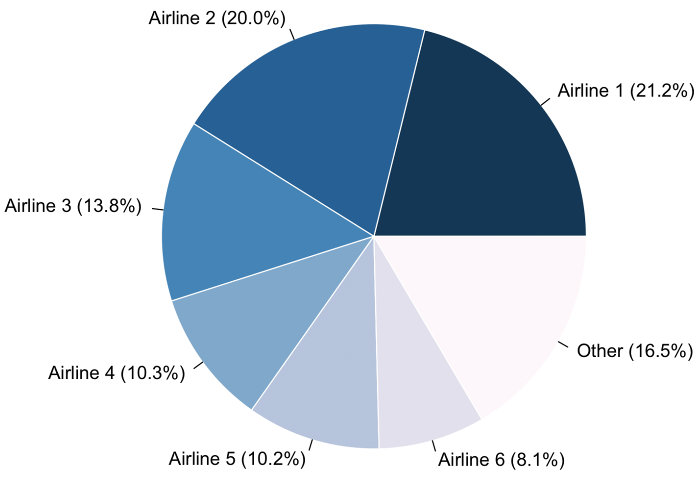

Figure 7. Share of flight operations by airline.

Figure 10 shows the scatterplot of ATOW and distance flown for the most frequent aircraft types within each wake turbulence category (medium turboprop, medium jet and heavy jet), suggesting that route length does not explain a significant part of the ATOW variability.

3.2 Predictive Performance Analysis

This subsection presents the predictive performance analysis of six algorithms: XGBoost, CatBoost, LightGBM, MLP (FastAI Tabular learner), SAINT, and an ensemble combining all models. Their performance is also compared to a baseline model that returns the MTOW as output. The study focuses on key metrics, including the best hyperparameters identified through optimization and the

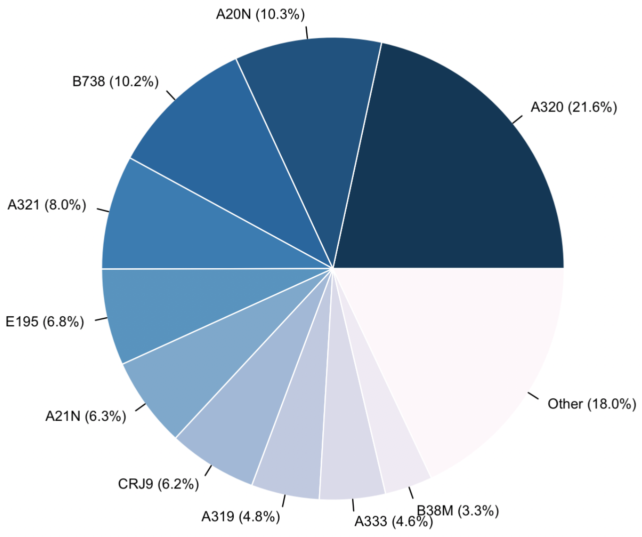

Figure 8. Share of flight operations by aircraft type.

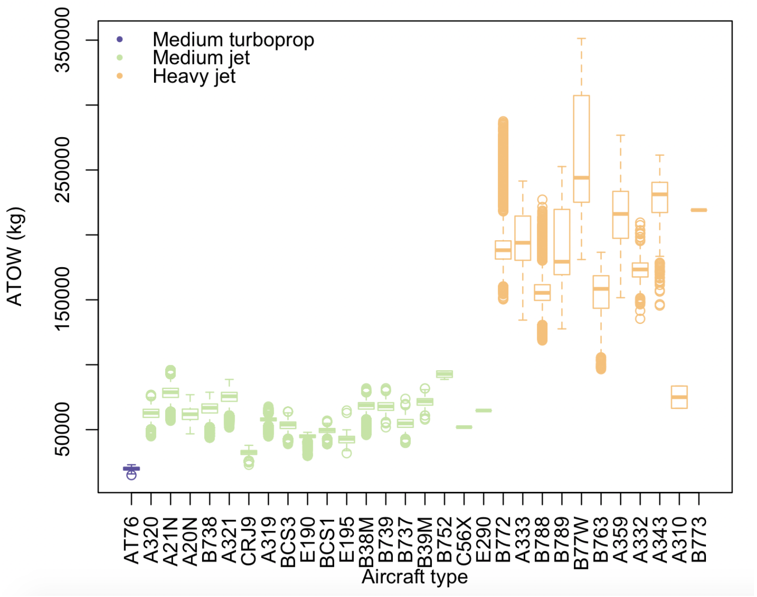

Figure 9. Boxplot of actual takeoff weight for each aircraft type.

estimated RMSE and MAPE for the test dataset. Notice that the test dataset is obtained after the 80-20 split done locally. We also present the actual RMSE for the final submission dataset provided by the Data Challenge to rank the participating teams.

The ensemble model combines predictions from the individual models, leveraging their complementary strengths to achieve improved predictive accuracy. As shown in Table 6, the ensemble model achieved the best test RMSE at 2210.37 and the lowest MAPE at 1.73%, both marked as the lowest

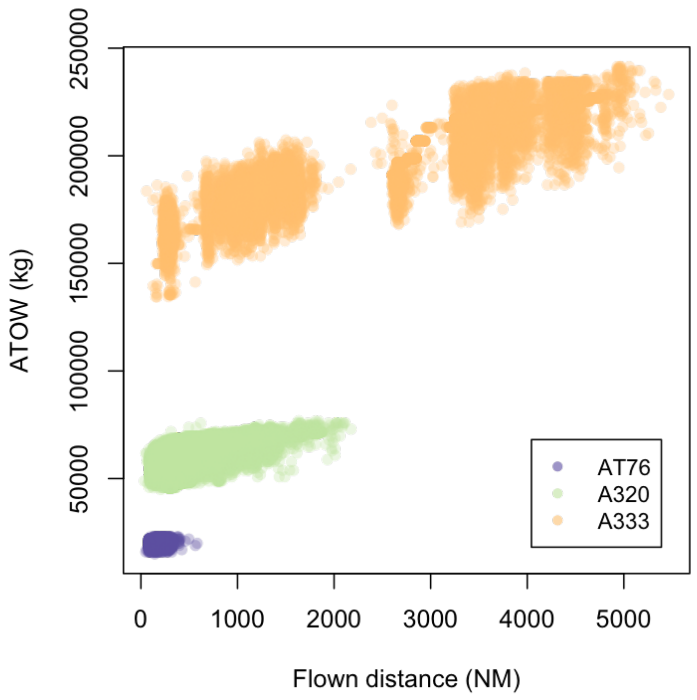

Figure 10. Scatterplot of the actual takeoff weight and distance flown by aircraft type.

errors among the models tested. The actual RMSE values for each algorithm align closely with their estimated test RMSE, indicating that the models generalize well to unseen data.

Among individual models, CatBoost achieved the lowest test RMSE at 2215.53 and MAPE of 1.75%, closely followed by LightGBM and XGBoost, which also demonstrated competitive RMSE and MAPE performance. Although the MLP model performed reasonably, it showed a slightly higher RMSE and MAPE compared to the gradient-boosted models, highlighting the effectiveness of GBDT models on tabular data for this task. As one can see, all of them significantly outperform the baseline.

Using our best model (Ensemble), we calculated the RMSE for each aircraft type, presented in Table 7. RMSE values vary significantly across different aircraft categories, indicating that model performance depends notably on aircraft type. For instance, larger aircraft such as B752, B77W, B789, and A359 exhibit higher RMSE values. Conversely, smaller aircraft types, including A319, AT76, and CRJ9, have lower RMSE values, suggesting the model predicts their takeoff weights more consistently.

3.3 Feature Importance Analysis

Figure 11 presents the top 20 feature importances obtained from the trained CatBoost model, which achieved the best individual performance among the models tested. These importances were computed using the `PredictionValuesChange` method [20], indicating the average change in prediction if a feature's value is altered. As expected, aircraft features are among the most critical predictors overall. We also notice a significant contribution of the airline feature, emphasizing the impact of different operational policies and preferences on fuel loading. Regarding the trajectory features, the actual distance flown and the average en route altitude were the most important predictors, high-

Table 6. Comparison of best hyperparameters, RMSE (estimated and actual), and Estimated MAPE for XGBoost, CatBoost, LightGBM, MLP, SAINT, and the Ensemble models. The best RMSE and MAPE values are highlighted in bold and blue.

Algorithm	Best Hyperparameters	Estimated Test RMSE	Actual RMSE	Estimated Test MAPE
Baseline (MTOW)	–	32,324.13	32,324.13	26.10%
MLP	`hidden_layers_size=[400, 300, 200], learning_rate=0.003`	3,864.1	3,864.2	4.54%
SAINT	`hidden_layers_size=[900, 400, 300], learning_rate=0.002`	3,921.3	3,921.5	4.63%
CatBoost	`learning_rate=0.01, reg_lambda=69.07, random_strength=16.35, depth=11, min_data_in_leaf=2, leaf_estimation_iterations=7`	2,215.53	2,224.69	1.75%
LightGBM	`learning_rate=0.01, reg_lambda=0.46, reg_alpha=0.17, min_child_weight=4, max_depth=13, colsample_bytree=0.60, subsample=1.0`	2,506.43	2,519.17	2.50%
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`	2,413.05	2,403.07	2.42%
Ensemble	`weighted average=[1.24e-23, 1.63e-23, 8.65e-01, 4.77e-22, 1.34e-01]`	**2,210.37**	**2,217.75**	**1.73%**

lighting the influence of the horizontal and vertical profiles on fuel burn. A complete description of the features used can be found at the code repository.

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 ATOW estimation model. The ablation study is a systematic experimental procedure used to evaluate the contribution of different components in a complex model or system. In machine learning, it involves removing or "ablating" specific parts of a model (such as feature groups or algorithmic components) and measuring the resulting change in performance. This methodology helps researchers identify which elements are most crucial to the model's effectiveness and which might be redundant. By quantifying the performance degradation that occurs when particular components are removed, ablation studies provide valuable insights into the relative importance of each component, guide future model refinements, and enhance interpretability. In our analysis, we employed this technique to assess the impact of different feature groups and model components on the predictive performance of our ATOW estimation model, using the RMSE as our evaluation metric.

The ablation process started with our optimal ensemble model as the baseline and 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

Table 7. Predictive performance of the ensemble model for each aircraft type.

Aircraft type	RMSE
A20N	1551.050
A21N	1733.565
A319	661.890
A320	1352.748
A321	1523.766
A332	3634.151
A333	4259.399
A343	4620.041
A359	5205.715
AT76	745.615
B38M	1765.090
B39M	1729.392
B737	1642.344
B738	1757.256
B739	1756.691
B752	11893.238
B763	3906.629
B772	4779.708
B77W	5456.838
B788	4712.170
B789	5158.968
BCS1	1415.357
BCS3	1492.246
CRJ9	819.703
E190	1169.980
E195	1247.490

the feature set, and the model was retrained and evaluated. Additionally, we assessed the impact of eliminating individual algorithms from the ensemble.

Table 8 presents the results of the ablation study, showcasing the RMSE values and percentage changes relative to the baseline model for each ablated configuration.

For clarity, the feature groups used in the ablation study correspond to specific subsections in the Feature Engineering section. The Temporal Features group refers to features described in Section 2.3.2, including hour of departure, day of the week, month, season, and other time-related attributes that capture cyclical patterns in flight operations. The Geographical Features group encompasses elements detailed in Section 2.3.4, including airport distances, altitude differences, regional classifications, and spatial relationships that provide geographical context to flight operations. Aircraft Characteristics features, as described in Section 2.3.6, include aircraft type parameters such as MTOW, OEW, fuel capacity, and physical dimensions that define the fundamental capabilities and constraints of each aircraft type. The Operational Features group combines elements described in Section 2.3.1 and in Section 2.3.2, encompassing flight trajectory features, vertical efficiency metrics, and other operational parameters that characterize how the aircraft was flown. Finally, the Meteorological Features group includes weather-related variables described within Section 2.3.1, such as tempera-

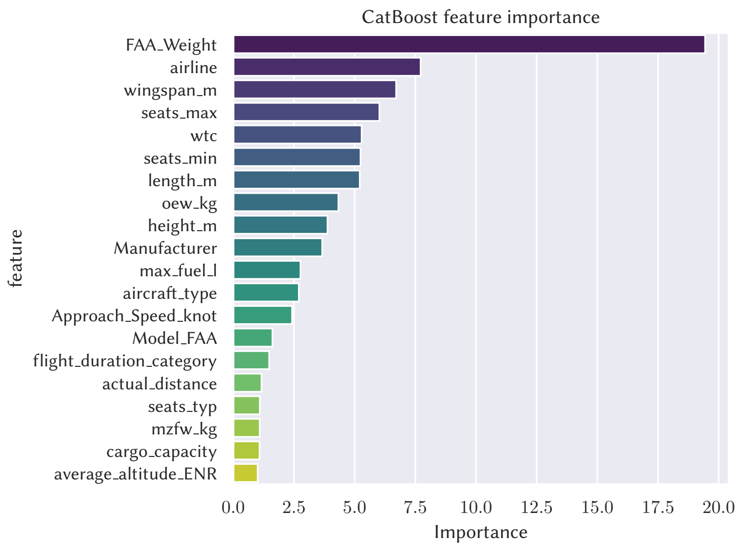

Figure 11. Top 20 feature importances for CatBoost.

ture, humidity, and wind components, which influence both flight planning and actual performance. This categorization enables a systematic assessment of how different feature domains contribute to the model's predictive performance, allowing us to quantify the relative importance of each feature group in accurately estimating aircraft takeoff weight.

Table 8. Ablation study results

Ablated Component	RMSE	Change in RMSE (%)
Baseline (Full Model)	2,210.37	0.00%
Temporal Features	2,354.97	+6.54%
Geographical Features	2,390.26	+8.14%
Aircraft Characteristics	2,266.96	+2.56%
Operational Features	2,759.18	+24.83%
Meteorological Features	2,249.29	+1.76%
XGBoost Algorithm	2,213.91	+0.16%
CatBoost Algorithm	2,969.49	+33.90%
LightGBM Algorithm	2,210.37	+0.00%
Neural Network	2,210.37	+0.00%
SAINT	2,210.37	+0.00%

The ablation study shows that the operational features processed from the ADB-S data exert the most substantial influence on model performance, with their removal resulting in a 24.83% increase in RMSE (from 2,210.37 to 2,759.18). This significant degradation in performance shows the crucial role of operational parameters in accurate ATOW prediction. Among the algorithmic components, the CatBoost algorithm demonstrates the highest individual impact when removed from the ensemble, as its removal leads to a substantial 33.90% increase in RMSE, given that it has the highest weight and the most contribution to the ensemble's predictive capability.

Geographical and temporal features also show notable importance, with their removal causing RMSE increases of 8.14% and 6.54%, respectively. This indicates that spatial and temporal context provides valuable information for ATOW prediction, though not as critical as operational characteristics. Aircraft characteristics contribute moderately to the model's performance, with their exclusion resulting in a 2.56% increase in RMSE.

Interestingly, meteorological features show relatively minimal impact, with only a 1.76% increase in RMSE when removed. This suggests that while weather conditions do influence ATOW predictions, their effect is less pronounced compared to other feature groups in our model.

In the context of model components, the results reveal varying degrees of importance among the different algorithms. While the removal of CatBoost significantly impacts performance, the XGBoost algorithm shows minimal effect (0.16% RMSE increase). The unchanged RMSE values (0.00% change) for LightGBM, Neural Network, and SAINT algorithms suggest these components might benefit from further optimization or may be redundant in the current ensemble configuration, and thus pruned.

These findings show the importance of careful feature engineering in ATOW prediction, particularly regarding operational parameters and geographical-temporal contexts. They also validate the efficacy of our ensemble approach, while highlighting potential areas for optimization, especially in the selection and integration of other different algorithmic components.

4. Conclusion

This study aimed to predict the aircraft's actual takeoff weight by employing machine learning techniques on large-scale open aviation data. We demonstrated that accurate predictive models can be developed for this essential operational parameter through extensive experimentation with various algorithms and feature engineering strategies. The results indicated that ensemble models combining CatBoost, LightGBM, XGBoost, and artificial neural networks achieved the best overall performance, with a root mean squared error of 2,217.75. Among individual models, the CatBoost algorithm was the strongest performer, achieving an RMSE of 2,224.69. The analysis shows the effectiveness of gradient-boosting methods in this domain.

The investigation into the importance of features revealed that aircraft type, airline and flight profile characteristics are essential in predicting takeoff weight. Incorporating detailed aircraft data significantly enhanced model performance, emphasizing the value of leveraging supplementary information beyond conventional aircraft type and category features. The ablation study confirmed that feature engineering was critical in improving accuracy, with distinct feature groups contributing uniquely to model effectiveness. These findings suggest that a rich feature set, including precise aircraft and operational characteristics, is essential for building reliable models in aviation data contexts.

The current approach, while promising, has several limitations. The reliance on purely data-driven models does not fully account for the aerodynamic and operational complexities inherent in aircraft weight estimation. Additionally, gaps in data coverage, mainly due to limitations in ADS-B signals, may affect model generalization across diverse geographical regions. Future research could address these challenges by incorporating analytical models that draw on aerodynamic principles and operational knowledge, integrating physics-based insights with machine learning. Such integration could provide physical constraints that enhance the reliability of the models. Additionally, exploring supplementary datasets, such as detailed aircraft specifications, meteorological data, and operational parameters, could further enrich feature sets, leading to more accurate predictions. Exploring advanced machine learning architectures and optimization methods could also contribute to refined predictions. Evaluating model performance across diverse operational conditions and extending the methodology to include other flight phases would enhance the applicability of these models in real-world air traffic management.

This work contributes to the broader aviation research community by demonstrating that machine learning methods can estimate aircraft takeoff weight using openly available data, achieving good performance. The developed models and methodologies lay a foundation for further advancements in aircraft weight estimation, with potential applications covering trajectory prediction and optimization and aviation's environmental impact assessment.

Acknowledgement

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

Author contributions

- Mayara C. R. Murça: Conceptualization, Formal Analysis, Investigation, Methodology, Supervision, Project Administration, Visualization, Writing (Original Draft), Writing (Review and Editing).
- Marcos R. O. A. Maximo: Conceptualization, Data Curation, Methodology, Software, Supervision, Project Administration, Visualization, Writing (Original Draft), Writing (Review and Editing).
- Joao P. A. Dantas: Data Curation, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization, Writing (Original Draft), Writing (Review and Editing).
- João B. T. Szenczuk: Data Curation, Formal Analysis, Investigation, Methodology, Visualization, Writing (Original Draft).
- Carolina R. Lima: Methodology, Software, Investigation, Writing the draft and reviewing.
- Lucas O. Carvalho: Data Curation, Formal Analysis, Investigation, Methodology, Writing (Original Draft).
- Gabriel A. Melo: Data Curation, Methodology, Software, Writing (Original Draft).

Open data statement

The data used for this research is openly available at https://github.com/PRC-Data-Challenge-2024/team_tiny_rainbow and https://doi.org/10.5281/zenodo.15069706.

Reproducibility statement

The open-source code for this research is available at https://github.com/PRC-Data-Challenge-2024/team_tiny_rainbow and https://doi.org/10.5281/zenodo.15069706.

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] Eurocontrol. *Vertical flight efficiency @ airports.* [Online; accessed October 2024]. URL: https://ansperformance.eu/efficiency/vfe/.

[17] S. Salgueiro, J. L. Huynh, and R. J. Hansman. "Aircraft takeoff and landing weight estimation from surveillance data". In: *AIAA SCITECH 2022 Forum.* 2022. DOI: 10.2514/6.2022-1307.

[18] L. Grinsztajn, E. Oyallon, and G. 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.

[19] T. Chen and C. 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.

[20] J. T. Hancock and T. M. Khoshgoftaar. “CatBoost for big data: an interdisciplinary review”. In: *Journal of big data* 7.1 (2020), p. 94.

[21] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T-Y Liu. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. In: *Advances in neural information processing systems*. 2017, pp. 3146–3154.

[22] A. A. Ibrahim, R. L. Ridwan, M. M. Muhammed, R. O. Abdulaziz, and G. A. Saheed. “Comparison of the CatBoost classifier with other machine learning methods”. In: *International Journal of Advanced Computer Science and Applications* 11.11 (2020).

[23] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. 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.

[24] Jeremy Howard and Sylvain Gugger. “fastai: A Layered API for Deep Learning”. In: *CoRR* abs/2002.04688 (2020). arXiv: 2002.04688. URL: https://arxiv.org/abs/2002.04688.

[25] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C. Bayan Bruss, and Tom Goldstein. “SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training”. In: *CoRR* abs/2106.01342 (2021). arXiv: 2106.01342. URL: https://arxiv.org/abs/2106.01342.

[26] T. G. Dietterich. “Ensemble methods in machine learning”. In: *International workshop on multiple classifier systems*. Springer. 2000, pp. 1–15.

[27] Mark J. van der Laan, Eric C. Polley, and Alan E. Hubbard. “Super learner”. In: *Statistical Applications in Genetics and Molecular Biology* 6 (2007), Article25. DOI: 10.2202/1544-6115.1309.

[28] Leo Breiman. “Stacked regressions”. In: *Machine Learning* 24.1 (July 1996), pp. 49–64. ISSN: 1573-0565. DOI: 10.1007/BF00117832. URL: https://doi.org/10.1007/BF00117832.