Automated Data-Driven Prediction on Aircraft Estimated Time of Arrival

Zhengyi Wang, Man Liang, Daniel Delahaye

Automated Data-Driven Prediction on Aircraft Estimated Time of Arrival

Zhengyi Wang[a,*], Man Liang[b], Daniel Delahaye[a]

[a] *Ecole National de l'Aviation Civile, Toulouse, France*
[b] *University of South Australia, Adelaide, Australia*

Abstract

4D trajectory prediction is the core element of the future air transportation system. It aims to improve the operational ability and the predictability of air traffic. In this paper, a novel automated data-driven framework to deal with the prediction of Estimated Time of Arrival (ETA) on the runway at the entry point of Terminal Manoeuvring Area (TMA) is introduced. The proposed framework mainly consists of data preprocessing and machine learning models. Firstly, the dataset is divided, analyzed, cleaned, and estimated. Then, the flights are clustered into partitions according to different runway-in-use (QFU). Several candidate machine learning models are trained and selected on the corresponding dataset of each QFU. Feature engineering is conducted to transform raw data into features. After that, the experiments are performed on real ADS-B data in Beijing TMA with nested cross validation. By comparing the prediction performance on the preprocessed and un-preprocessed datasets, the results demonstrate that the proposed data preprocessing is able to improve the data quality. It is also robust to outliers, missing data, and noise. Finally, an ensemble learning strategy named stacking is introduced. Compared to other individual models, the stacked model has a more complex structure and performs best in ETA prediction. This fact reveals that the framework proposed in this study could make accurate and reliable ETA predictions.

Keywords: Air Traffic Management, 4D Trajectory Prediction, Estimated Time of Arrival, Data Mining, Machine Learning,

[☆] This paper is an extension of work originally reported in SESAR Innovation Days 2018 [1].

[*] Corresponding author
Email address: zhengyi.wang@alumni.enac.fr (Zhengyi Wang)

1. Introduction

4D trajectory and flight information are crucial factors for the Trajectory Based Operation (TBO) and Collaborative Decision Making (CDM). Highly accurate 4D Trajectory Prediction (TP) capability is the cornerstone of TBO deployment. Through accurate 4D TP, the air traffic operational efficiency could be improved. The flight cost and adverse environmental impact could also be lowered. More importantly, the workload of Air Traffic Controllers (ATCO) could be alleviated, which means that the maximum Air Traffic Management (ATM) capacity could be augmented [2].

Previous studies on TP could be divided into model-driven methods and data-driven methods. Classical model-based TP methods made the ideal assumptions about the motion of the aircraft, the atmospheric environment, and the flight performance, along with either parametric or physics-based trajectory models. They generally do not take the intersections between different trajectories and the real Air Traffic Control (ATC) human behavior factors into account. In addition, lacking sufficient data support, computation resources, and learning ability, the model-based TP method is much less effective when facing massive real-time data in large-scale ATM systems. To overcome the drawbacks of model-driven approaches, nowadays, the focus of 4D TP has been gradually shifted to data-driven approaches.

The Data-driven AiRcraft Trajectory prediction research (DART) project is one of the recently launched research projects supported by SESAR joint undertaking, aiming to explore the applicability of data-driven approaches to the ATM domain [3]. As pointed by DART, data-driven techniques are able to train appropriate models from all relevant and actual historical data with no or few prior assumptions and few requirements for data quality. Compared to classical model-driven approaches, contextual features can be extracted, including ATC information, meteorological condition, human factors, which will be beneficial in modeling the ATM socio-technical system and taking operational constraints into account.

Machine learning models are the most prevailing techniques in data-driven approaches. They have solid and widely accepted mathematical foundations and can provide insights on the air traffic dynamics [4, 5, 6, 7, 8, 9, 10, 11, 12]. However, most prediction models have difficulties in handling 4D TP scenarios with high-complexity, multi-dimensions and high-nonlinearity. A well-performed prediction framework is supposed to deal with outliers, missing, or noisy data [13]. In addition, some studies require complicated preprocessing steps, which are trade-offs between prediction performance and computational efficiency, which need to be weighed carefully. Furthermore, most current models lack generalizability and automaticity. They are only applicable to one or a few flights, aircraft types, or departure/arrival procedures. If the problem is extended to other flights, airports, airspaces, or scenarios, the defined model architectures and parameters need to be greatly modified. Last but not least, the data used in some researches are not openly shared due to security reasons and business interests. Some other flight data sources, e.g., Quick Access Recorder (QAR) data, Flight Data Recorder (FDR) data, are not possible to implement real-time 4D TP.

The objective of this paper is to improve ETA prediction accuracy by means of data preprocessing and machine learning. The trade-off between computational efficiency and prediction performance is also studied. Based on open historical ADS-B data of BCIA, 6 machine learning models are applied to predict the Estimated Time of Arrival (ETA) of flights on the runway at the entry point of Terminal Manoeuvring Area (TMA). Due to complex traffic patterns, meteorological conditions, ATC command, and human behavior, the entry point is one of the most challenging and important areas for ETA prediction. Besides, the information at the entry of TMA is of great interest to the controllers (ATCO). Several preliminary efforts have been made by authors. For example, [12] introduced an ETA trajectory prediction framework, including clustering-based preprocessing and Multi-Cells Neural Networks (MCNNs). However, this model is not fully automated. Firstly, only flights with the same magnetic orientation of the runway in use (QFU) can be handled by the model. If flights with multiple QFUs are taken into account, the trajectories will overlap and mix, which will greatly bring difficulties for the clustering algorithm to extract meaningful traffic patterns. Secondly, hyperparameters need to be retuned if we generalize the model to other TMAs. As an extension of [1], this paper refines and optimizes the preprocessing steps. In the prediction section, more prevailing machine learning models are compared. Moreover, a more powerful ensemble learning strategy named stacking is utilized in this study. As a result, a more powerful ETA prediction framework is proposed.

The main contributions of this paper could be listed as the following:

1. A refined preprocessing method is applied to handle more complicated traffic patterns. It is robust and generalizable. It is capable of processing 4D trajectory data of all landing flights in TMA, even with outliers, missing points, and noise. Besides, the approach could be extended to other TMAs, without manually tuning the hyperparameters. In addition, the preprocessing step is supposed to improve the prediction performance with very few computational complexities.
2. An automated ETA prediction model is developed to handle routine traffic at the entry of TMA. A comparative study is conducted for selecting the base learners from candidate machine learning models. Furthermore, base learners and stacking strategies are utilized to deal with the highly-dense arrival trajectories in TMA. The prediction framework is more complex, effective, and accurate than previous models.

2. Data Preprocessing

2.1. Data preparation

2.1.1. Airport and TMA

Beijing Capital International Airport (BCIA, ICAO: ZBAA) is selected as the study case. It is one of the busiest airports in the world, with three parallel runways: 18R/36L, 18L/36R, and 01/19. In 2017, BCIA was the airport with the most irregular flights in China, reaching 87,300 [14]. According to the China Electronic Aeronautical Information Publication (EAIP) [15], the lateral limits of Beijing TMA are plotted as blue boundaries in Fig. 1. To illustrate the range of Beijing TMA more clearly, we draw green concentric circles with radius from 10Nm to 70Nm.

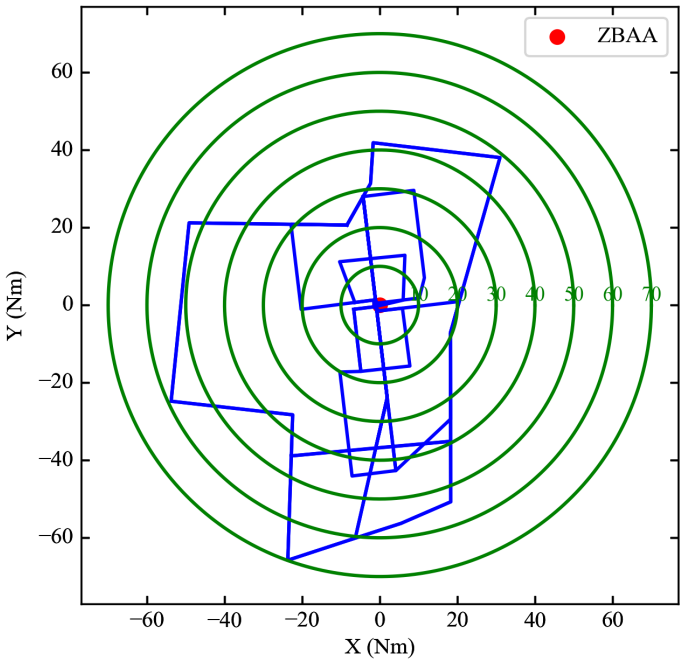


Figure 1: the lateral limits of Beijing TMA

2.1.2. Data description

The data source of this study is ADS-B data, which is easily accessible and can provide accurate real-time reports of aircraft's information. The dataset used throughout this study includes ADS-B records in July 2017 that belong to Beijing TMA. Since the range of Beijing TMA is relatively small, Geographic Coordinate System (GCS) can be projected into the Projection Coordinate System (PCS). Each record of ADS-B data contains the following information:

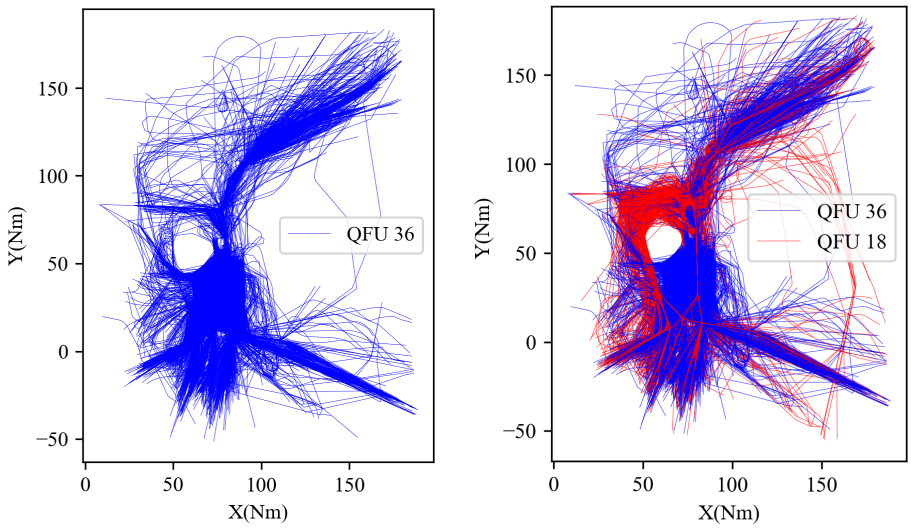
- Type of operation (departure/arrival),
- Runway in use,
- Coordinated Universal Time (UTC) timestamp t,
- Flight number i,
- Position (X, Y, Z),
- Heading H,
- Horizontal ground speed V_h
- Vertical ground speed V_v

Here, each record with flight number i belongs to the same flight. The collection of all records for that flight forms the trajectory T_i, $i = 1, ..., n$, where n is the total number of trajectories in the raw dataset. In this dataset, $n = 12775$.

2.1.3. Data volume

Unlike the dataset used in our previous work [12] that only considered a certain traffic operational direction, the dataset used in this study consists of all operational directions of arrival flights in Beijing TMA. Note that the runway in use information about each flight can be obtained from the flight plan.

Raw trajectories in these two studies are compared in Figure 2. In the previous model, the clustering algorithm is suitable for handling trajectories of a specific QFU, for example, QFU-36, see Figure 2a. However, It is difficult to handle very complex and overlapping trajectories of multiple QFUs. The new model used in this study will overcome the shortcoming of the previous model. It is able to handle multiple runway-in-use directions at the same time, both QFU-36 and QFU-18, see Figure 2b. These raw trajectory data are much messier, and it is difficult to observe obvious clusters. These flights can be seen as negative contributing factors to the prediction task. Thus, it is a challenge to conduct an accurate 4D trajectory prediction in Beijing TMA with classic approaches.



(a) Trajectories of QFU-36 (b) Trajectories of QFU-36 and QFU-18

Figure 2: Trajectories in the dataset of previous study and this study

2.1.4. Data split

The trajectories in the raw dataset are randomly split into two parts. Both parts account for 50%. The first part is firstly preprocessed with the following steps introduced in this section. The objective of the first part is to select basic learners. Another 50% dataset is used for experiments on the stacked model and other machine learning models.

2.2. Data cleaning

The raw dataset is relatively messy. It even contains some unusual trajectories and missing values. These outliers and inconsistent values can lead to misdirected predictions. Therefore, before making predictions, cleaning the raw data is necessary. Data cleaning refers to identifying incomplete, incorrect, inaccurate, or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data. In this paper, the raw dataset is firstly cleaned according to the following criteria:

2.2.1. Data receiving problem

When integrating multiple data sources, different records of the same trajectory may be provided with the same timestamp. In this case, we keep one of them. Besides, trajectories with very few recording points were eliminated from the dataset. 50 points are set to be the threshold.

2.2.2. The landing points

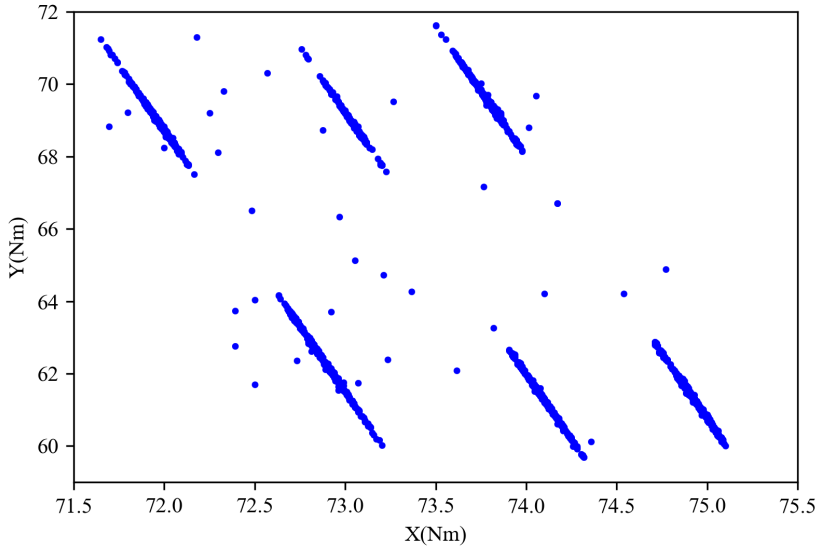
In order to get the Actual Time of Arrival (ATA), the first grounded point of each trajectory is regarded as the landing point. The landing points of all trajectories are shown in Figure 3a. The landing points can be clustered into 6 main partitions according to configuration and region of runways through analysis. Most landing points are along the runway. Nevertheless, due to the unstable receiving quality of ADS-B data, some trajectories are not complete. Few landing points are out of runway. These incomplete trajectories account for nearly 2%, which have to be filtered. Overall, the data validity rate reaches over 98%, which ensures the quality of the dataset. The filtered landing points are shown in Figure 3b. The 6 partitions were labeled with magnetic orientations of runway-in-use. Centroids of each cluster are also calculated and plotted. The results demonstrate that all incomplete trajectories can be successfully filtered.

2.2.3. Transit time in TMA

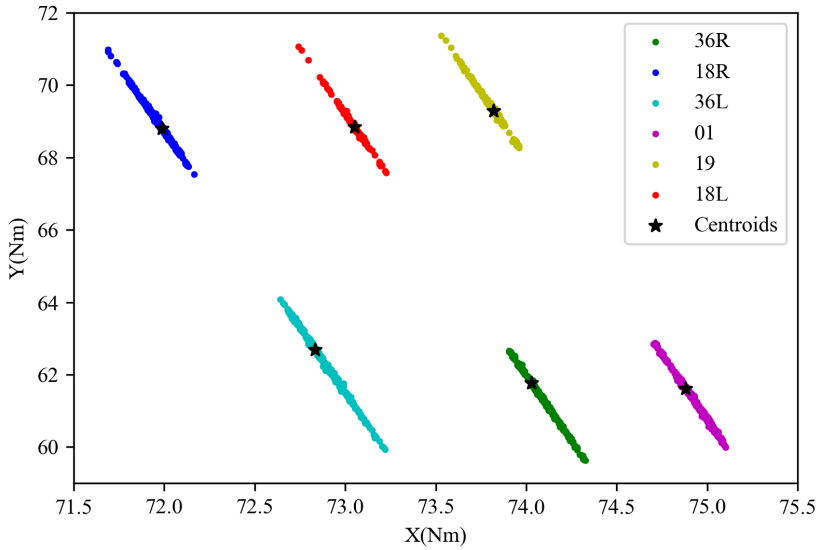
The transit time of an aircraft represents the elapsing time from entering the TMA to landing. After data cleaning, the distribution of transit time of flights in terms of QFUs is respectively plotted in Figure 4. In each subfigure, transit times are sorted in ascending order. For each magnetic orientation of landing, It can be observed that very few flights have long transit time, accounting for about 1%. The trajectories of these flights are further visualized in Figure 5. It can be seen that most trajectories include holding patterns, large vectors, go-around procedures. Though ETA prediction of outliers is also an important topic, this research only focuses on typical behavior. In addition, the data size of outliers is not enough for using data-driven techniques. As negative contributing factors for ETA predictions, these unrepresentative flights are regarded as stochastic and irregular flights that need to be filtered. In addition, for new inputs, if the predicted transit time exceeds the threshold (1% longest transit time of corresponding QFU, listed in table 1). The prediction results cannot reflect the actual performance and should be neglected. Figure 6 portrays the remaining 99% trajectories in terms of each QFU. It can be seen that these trajectories are much more regular after removing noises. In addition, most entry points of QFU-18R, 18L, and 19 are aggregated on the top, while the entry points of QFU-36L, 36R, and 01 are concentrated on the top. This fact also proves that landing points clustering can improve data quality. In each QFU, most trajectories follow similar patterns, which could lead to better predictability.

2.2.4. Area of study

As illustrated in Figure 1, the nearest distance between entry points of TMA and runways of BCIA is approximately 20 to 25 Nm, which is the maximum radius where all flights are present. Therefore the model can take them all into account with this choice of area. The trajectory points between 2 circles with a radius



(a) Landing points before filtering



(b) Landing points after filtering

Figure 3: Landing points before and after filtering in terms of QFUs in Beijing TMA

of 20 NM and 25 NM are kept. In addition, in order to introduce more valuable time-dependent features in the following section, we also keep a few points before each trajectory point. According to the experience, the previous 4 points are set to be kept.

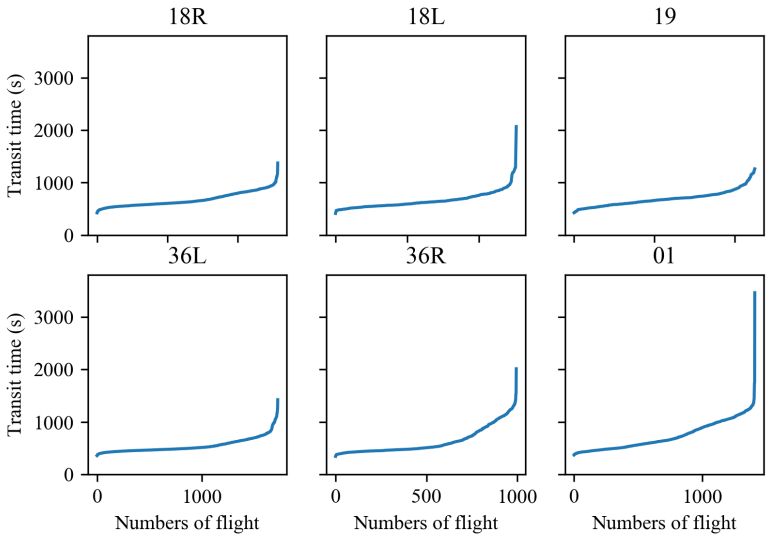


Figure 4: Distribution of transit time of different QFUs in Beijing TMA

Table 1: Transit time threshold of different QFUs in Beijing TMA

QFU	Transit time threshold (s)
18R	1068
18L	1286
19	1190
36L	1080
36R	1340
01	1336

2.2.5. Missing points estimation

The update interval is referred to as the time interval between succeeding reports. Through analysis of the ADS-B dataset, nearly 98.9% update frequency is within 10 seconds, but only 72.5% update frequency is within 2 seconds. To estimate the missing information and to enrich the dataset, piecewise linear interpolation is employed for estimation.

Given $N+1$ real numbers y_i, $0 \leq i \leq N$, and $N+1$ distinct real numbers $x_0 < x_1 < ... < x_N$, we consider the N linear curves $l_i(x) = a_i x + b_i$ on the intervals $[x_i, x_{i+1}]$ for $i = 0, ..., N-1$. Each $l_i(x)$ has to connect two points (x_i, y_i) and (x_{i+1}, y_{i+1}) according to the following equations:

$$y_i = a_i x_i + b_i \tag{1}$$

$$y_{i+1} = a_i x_{i+1} + b_i \tag{2}$$

For all attributes in the dataset, the update interval is upsampled to 1 second.

2.3. Feature engineering

2.3.1. Feature selection

In our previous research [1], for timestamp t, only the information of one trajectory point $(X_t, Y_t, Z_t, H_t, V_{h,t}, V_{v,t})$ is contained in each input. Although fewer numbers of features can accelerate the training process of the prediction model, they also increase the uncertainty and may lead to satisfactory results. More specifically, there are no specific rules to exactly calculate the position, heading and speed of a flight at a given time. Only one trajectory point is difficult to represent the current state. To this end, we introduce features at each trajectory point and its previous M points. The choice of M is a trade-off, as the longer time

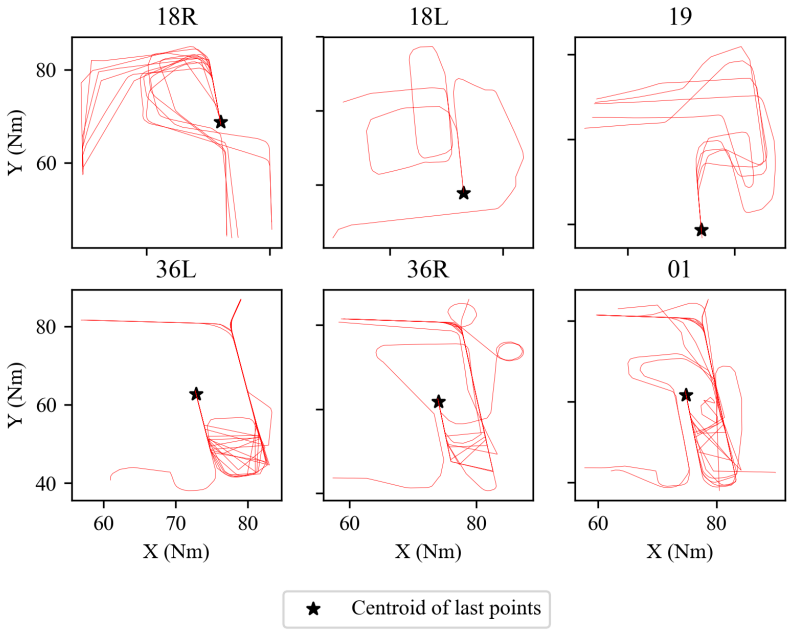


Figure 5: Trajectories of flights with top 1% longest transit time of different QFUs in Beijing TMA

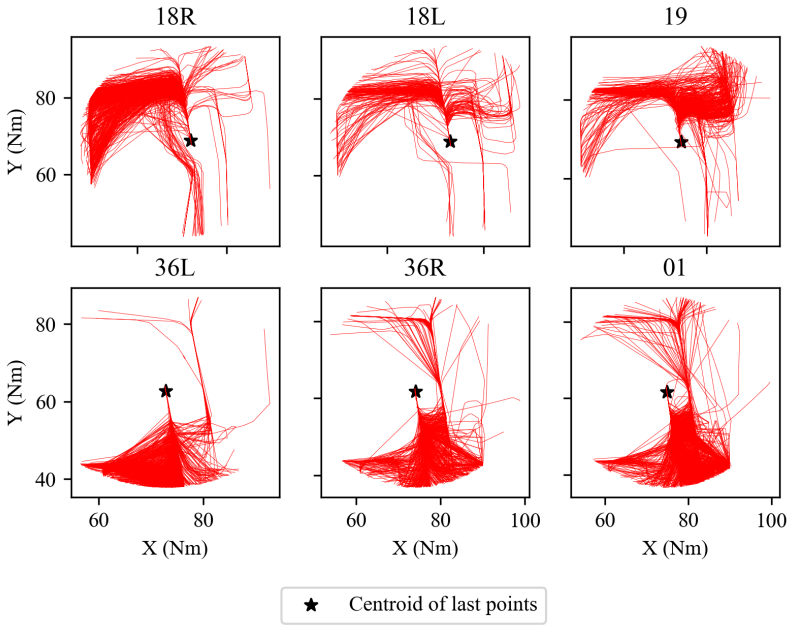


Figure 6: with other 99% trajectories of different QFUs in Beijing TMA

passes, the less important the information becomes. According to experience, M is set to be 4. Therefore, the feature vector becomes $(X_{t-4}, Y_{t-4}, Z_{t-4}, H_{t-4}, V_{h,t-4}, V_{v,t-4}, ..., X_t, Y_t, Z_t, H_t, V_{h,t}, V_{v,t})$.

2.3.2. Feature scaling

The selected features have different units, for instance, Nm, Knot, Deg, making their magnitudes different. If left alone, each feature would be directly put into the prediction model, neglecting their units. This will make the objective functions of most machine learning models work less effectively. To this end, we scale the features for all attributes. Given attribute vector $\mathbf{a}$, the transformation is defined as follows:

$$\mathbf{a}^* = \frac{\mathbf{a} - \text{Median}(\mathbf{a})}{\text{IQR}(\mathbf{a})} \tag{3}$$

where Median($\mathbf{a}$) returns the median value of $\mathbf{a}$. IQR($\mathbf{a}$) stands for the interquartile range of $\mathbf{a}$. That is, the difference between first quartile Q_1 and third quartile Q_3 of $\mathbf{a}$. This scaler is also called a robust scaler, as it is robust to outliers.

3. ETA Prediction models

The goal of this section is to predict the ETA of aircraft at the entry of TMA. This is a regression problem. The idea is to approximate the mapping function h from input vector $\mathbf{x} \in \mathbb{R}^D$ in the input space $\mathcal{X}$ to the corresponding output vector $t \in \mathbb{R}$ in the output space $\mathcal{Y}$:

$$t = h(\mathbf{x}) + \epsilon \tag{4}$$

where ϵ is random noise.

The prediction framework is illustrated in Figure 7. Firstly, we propose N_m prevailing machine learning models as candidates. On the first 50% dataset, nested cross validation is conducted on each model to tune the hyperparameters and to evaluate the prediction performance. To decrease the computational burden, the number of base learners M_m is less than that of candidate machine learning models N_m. The first M_m machine learning models with the smallest prediction errors are selected as base learners. The hyperparameters of these base learners are also tuned to be optimal. Then, M_m base learners are ensembled by a specific strategy called stacking, which will be introduced detailly in section 3.3. It is a great challenge to select the hyperparameters for each individual model in the stacked model. That's why we conduct this process when selecting the base learners. The stacked model is trained by the rest 50% dataset and then make predictions of ETA. A comparative study is also performed on this dataset.

3.1. Machine learning candidate models

In order to get the optimal prediction performance, we firstly propose N_m prevailing machine learning models as candidates. They are described in table 2 and $N_m = 6$. Then, a comparative study is conducted on the dataset generated in section 2 to select M_m base learners. In order to increase computational efficiency without degrading prediction performance, M_m is set to 3.

Table 2: Machine learning candidates models

Type	Algorithm
Linear	Multiple Linear Regression (MLR)
Non-Linear	Feed-Forward Neural Networks (FFNNs) K-Nearest Neighbors (KNN)
Ensemble	Gradient Boosting Machine (GBM) Random Forests (RF) ExtraTrees (ET)

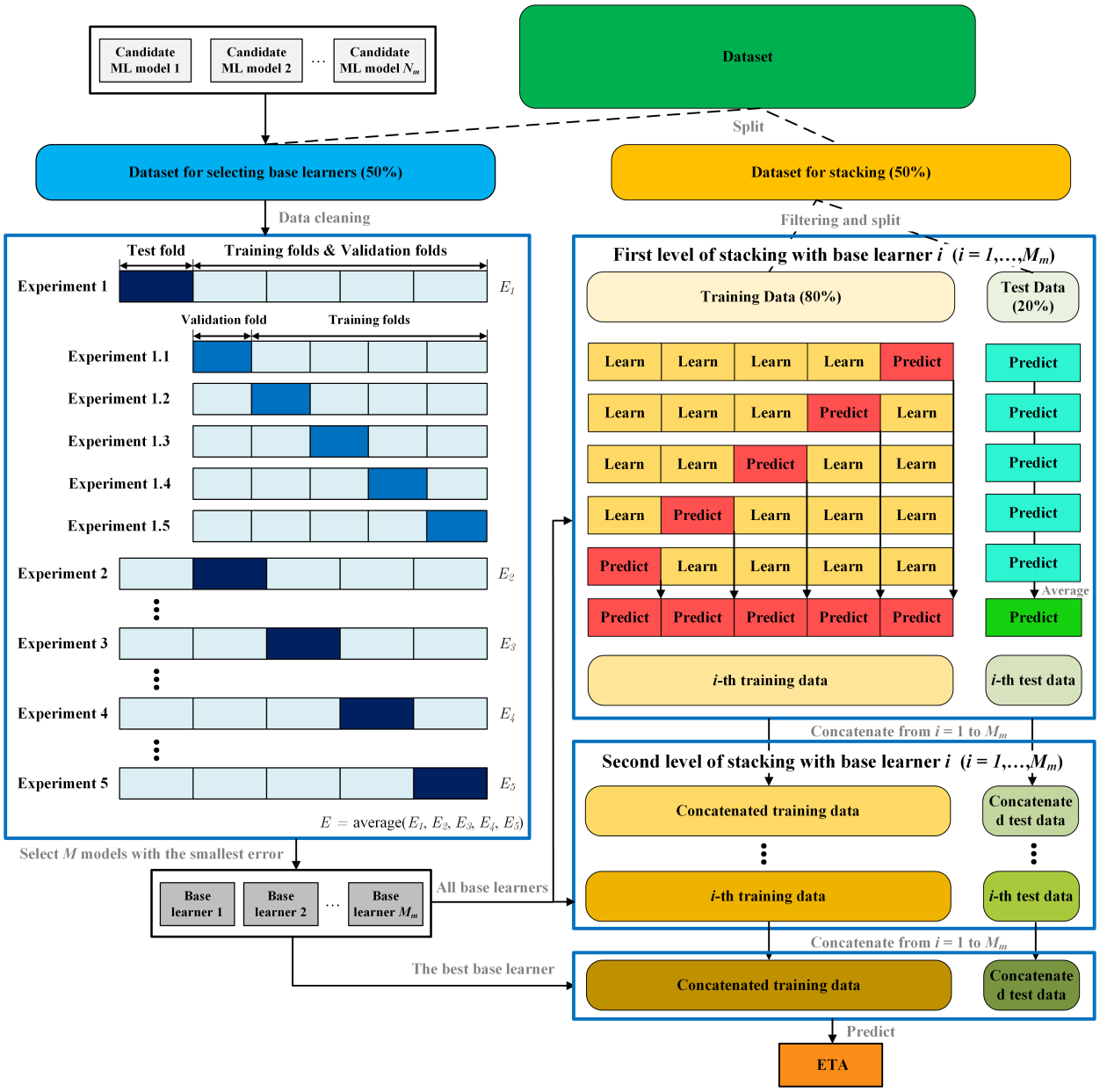


Figure 7: ETA prediction framework based on data cleaning and machine learning

3.1.1. Multiple Linear Regression

Multiple Linear Regression (MLR) is one of the most classical machine learning models. It aims to estimate the function $f(\cdot)$ under linear assumption. Given multiple independent variables $x_0, x_1, ..., x_D$, where $x_0 = 1$, then the output can be estimated as:

$$y(\mathbf{x}; \mathbf{w}) = \sum_{d=1}^{D} w_d x_d + w_0 \tag{5}$$

where $\mathbf{w} = (w_0, w_1, ..., w_D)^T$ are weights.

To prevent overfitting, the error function is chosen as the modified form and calculated on the training set $\mathcal{S} = \{(\mathbf{x}_1, t_1), ..., (\mathbf{x}_N, t_N)\}$:

$$L(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(\mathbf{x}_n, \mathbf{w}) - t_n)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{6}$$

Where λ is the regularization term. This particular case with a quadratic regularizer is also known as ridge regression [16].

The optimal $\mathbf{w}^*$ is given by the least-squares method:

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X} + \lambda I_D)^{-1}\mathbf{X}^T\mathbf{t} \tag{7}$$

where $\mathbf{t} = (t_1, ..., t_N)^T$ and

$$\mathbf{X} = \begin{pmatrix} 1 & \mathbf{x}_1^T \\ 1 & \mathbf{x}_2^T \\ \vdots & \vdots \\ 1 & \mathbf{x}_N^T \end{pmatrix}$$

3.1.2. Feed-Forward Neural Networks

In these experiments, a specific class of NNs is introduced to approximate the function $f(\cdot)$, referred to as Feed-Forward Neural Networks (FFNNs), which is one of the most quintessential deep learning models. FFNNs have an input layer, L hidden layers $h^{(1)}, ..., h^{(L)}$ ($L \geq 1$), and an output layer. Given input vectors $\mathbf{x} = (x_1, ..., x_D)^T$, the output $y(\mathbf{x}, \mathbf{w})$ is expressed as follows:

$$h^{(1)} = \Phi^{(1)}({\mathbf{w}^{(1)}}^\top \mathbf{x}) \tag{8}$$

$$h^{(l)} = \Phi^{(l)}({\mathbf{w}^{(l)}}^\top h^{(l-1)}), \quad l = 2, ..., L \tag{9}$$

$$y(\mathbf{x}, \mathbf{w}) = \Psi({\mathbf{w}^{(L+1)}}^\top h^{(L)}) \tag{10}$$

Where $\mathbf{w}^{(1)}$, $\mathbf{w}^{(l)}$ ($l = 2, ..., L$), $\mathbf{w}^{(L+1)}$ are respectively weights assigned to the connections between input layer and first hidden layer, between $(i-1)$-th hidden layer and i-th hidden layer, and between L-th hidden layer and output layer. $\Phi^{(l)}$ is the activation function applied to the weighted output of the i-th layer of NNs. Ψ is the function applied to the weighted sum of the activations of the last hidden layer. Note that if $L = 1$, DFNNs degenerate to shallow neural networks with 1 hidden layer. Frequently used Φ including Rectified Linear Unit (ReLU) function, hyperbolic tangent (tanh) function and sigmoid function, which are respectively defined as follows:

$$\Phi_{\text{ReLU}}(x) = \max(0, x) \tag{11}$$

$$\Phi_{\tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{12}$$

$$\Phi_{\text{sigmoid}}(x) = \frac{1}{1 + e^{-x}} \tag{13}$$

Ψ is the identity function:

$$\Psi(z) = z \tag{14}$$

The error function on a training set T is defined as follows:

$$L_i(\mathbf{w}) = \sum_{(\mathbf{x},t)\in T}(y_i(\mathbf{x}, \mathbf{w}) - t)^2 + \lambda\sum_l \left\|\mathbf{w}^{(l)}\right\|_2^2 \tag{15}$$

where λ is the regularization coefficient. A properly selected λ can avoid overfitting.

The backpropagation method was used to compute the gradient of the error function, and Adam [17] was used to minimize the error between our predictions of ETA and the ATA. We used an algorithm commonly known as *reduce the learning rate on plateau* to reduce the learning rate when a metric stopped improving [18]. The learning rate can be adaptively adjusted by this algorithm and need not be tuned. In this paper, the initial learning rate is set to be 0.1. The learning rate will be reduced by a factor of 0.5 once there is no improvement of the error on the training set for 10 epochs. After the learning rate has been reduced, wait for 10 epochs before resuming normal operation. The lower bound on the learning rate is 10^{-4}. All layers of each NNs are fine-tuned with 1000 epochs of training. All these methods were implemented in PyTorch [19].

3.1.3. K-Nearest Neighbors

K-Nearest Neighbors (KNN) [20] is a lazy learning algorithm, in which the prediction is made locally on the delayed training set. In the regression analysis, KNN approximates the target by local interpolation of the nearest neighbors in the training set. The pseudocode of KNN is given in Algorithm 1.

Algorithm 1 KNN for regression

Input:
$\mathbf{X}$: Input of the training set, in $\mathbb{R}^{N\times D}$
$\mathbf{Y}$: Output of the training set, in $\mathbb{R}^{N\times 1}$
$\mathbf{x}$: New input vector, in $\mathbb{R}^{D}$
Output:
y: Output of $\mathbf{x}$, in $\mathbb{R}$

procedure KNN_Regression($\mathbf{X}$, $\mathbf{Y}$, $\mathbf{x}$)
  for $n = 1$ to N **do**
    Compute distance $d(\mathbf{X}_n, \mathbf{x})$
  end for
  $N_K \leftarrow K$-NN of $\mathbf{x}$ according to $\{d(\mathbf{X}_n, \mathbf{x})\}_{1\le n\le N}$
  $y \leftarrow \sum_{n\in N_K} \mathbf{Y}_n w(d(\mathbf{X}_n, \mathbf{x}))$
  return y
end procedure

The distance metric of KNN describes feature similarity. It is commonly set as Minkowski distance:

$$d(\mathbf{m}, \mathbf{n}) = \left(\sum_{j=1}^{D} \left| m_j - n_j \right|^p \right)^{1/p} \tag{16}$$

where $\mathbf{m}, \mathbf{n}$ are vectors in $\mathbb{R}^D$, p is the order. For example, when $p = 2$, the distance corresponds to the Euclidean distance. $w(\cdot)$ is the weight function. This function has two frequently used types: uniform weighting and inverse distance weighting.

Uniform weighting function assuming all neighbors are weighted equally:

$$w(d(\mathbf{X}_n, \mathbf{x})) = \frac{1}{card(N_K)}, \quad \forall n \in N_K \tag{17}$$

where $card(\cdot)$ assigns the cardinality of a set.

Inverse distance weighting function weighs neighbors by the inverse of their distance:

$$w(d(\mathbf{X}_n, \mathbf{x})) = \frac{1/d(\mathbf{X}_n, \mathbf{x})}{\sum_{m\in N_K} 1/d(\mathbf{X}_m, \mathbf{x})}, \quad \forall n \in N_K \tag{18}$$

3.1.4. Gradient Boosting Machine

Gradient Boosting Machine (GBM) is a famous ensemble learning method [21]. It can be viewed as iterative functional gradient descent algorithms, which produces a prediction model formed by an ensemble of weak prediction models, typically tree-based models. Gradient Boosting Decision Tree (GBDT) is a frequently used boosting type of GBM. Other methods include Dropouts meet Multiple Additive Regression Trees (DART) [22], Gradient-based One-Side Sampling (GOSS) [23]. The advantage of GBM is that even missing data can be handled. Besides, all differentiable loss function can be applied, making the model more flexible and more resistant to noise.

Given a training set $\{(\mathbf{x}_1, t_1), ..., (\mathbf{x}_N, t_N)\}$ and expected number of weak learners M, where $\mathbf{x}_n \in \mathbb{R}^D$, GBM learns weak learners incrementally. Initializing the model with a differentiable loss function $L(\cdot, \cdot)$:

$$F_0(\mathbf{x}) = \arg\min_{\gamma} \sum_{n=1}^{N} L(t_n, \gamma) \tag{19}$$

Then the m-th weak learner $h_m(\cdot)$ is trained by set $\{(\mathbf{x}_1, r_{1m}), ..., (\mathbf{x}_N, r_{Nm})\}$, where $m = 1, ..., M$, and

$$r_{nm} = -\left[\frac{\partial L(t_n, F(\mathbf{x}_n))}{\partial F(\mathbf{x}_n)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \quad n = 1, ..., N \tag{20}$$

r_{nm} is also known as pseudo-residuals. Then the prediction model is updated by the following equation:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \gamma_m h_m(\mathbf{x}) \tag{21}$$

where the multiplier γ_m is computed by:

$$\gamma_m = \arg\min_{\gamma} \sum_{n=1}^{N} L\left(y_n, F_{m-1}(\mathbf{x}_n) + \gamma h_m(\mathbf{x}_n)\right) \tag{22}$$

3.1.5. Random Forests

Random Forests (RF) [24] are a popular tree-based ensemble learning method. They are a combination of tree predictors such that each tree in the forest depends on the values of a random vector sampled independently and with the same distribution. By combination of weak learners, a stronger learner is generated.

As an ideal candidate and an extended variant for bootstrap aggregating (bagging) algorithm, the idea in RF is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance significantly. Because of the law of large numbers, overfitting is seldom seen in RF with a sufficient data. Based on tree models, RF is also robust to outliers.

The pseudo-code of RF is given in Algorithm 2 [25].

3.1.6. ExtraTrees

ExtraTrees (ET) is very similar to RF, in that it is also a popular tree-based model and belong to ensemble learning algorithms [26]. There are essentially 2 main differences:

1. RF performs bagging, but ET uses all the training samples to train each decision tree. In other words, each decision tree is trained on the same training samples.
2. RF selects the "best split" in a random subset. However, ET makes a small number of randomly chosen split value, then yield the best split from all the randomly generated splits. Such splitting makes ET more random than RF.

For better comparison with RF, the special splitting technique of ET is described in Algorithm 3.

Algorithm 2 RF for regression

Input:
$\mathbf{X}$: Input of the training set, in $\mathbb{R}^{N\times D}$
$\mathbf{Y}$: Output of the training set, in $\mathbb{R}^{N\times 1}$
$\mathbf{x}$: New input vector, in $\mathbb{R}^{D}$
Output:
y: Output of $\mathbf{x}$, in $\mathbb{R}$

procedure RF_Regression($\mathbf{X}$, $\mathbf{Y}$, $\mathbf{x}$)
  for $i = 1$ to L **do**
    Draw a bootstrap sample D_b with replacement of size N_b from training data $D = (\mathbf{X}, \mathbf{Y})$
    Build regression tree T_i on D_b, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{\min}$ is reached:

    1. Select m variables at random from the D variables.
    2. Pick the best variable among the m.
    3. Split the node into two daughter nodes.

  end for
  $y \leftarrow 1/L \sum_{i=1}^{L} T_i(\mathbf{x})$
  return y
end procedure

Algorithm 3 ET splitting

Input:
S: The local learning subset
Output:
A: split $[a < a_c]$ or void

procedure Split_A_Node(S)
  if $|S| < n_{\min}$ **or** all attributes are constant in S **or** The output is constant in S **then**
    return
  else
    Select K attributes $\{a_1, .., a_k\}$ among all non constant candidate attributes in S.
    for $i = 1$ to K **do**
      $a^S_{i,\max} = \max_{a\in S}(a),\ a^S_{i,\min} = \min_{a\in S}(a)$
      Draw a random split-point a_c uniformly in $[a^S_{i,\max}, a^S_{i,\min}]$
      $s_i = [a < a_c]$
    end for
    $\text{Score}(s^*, S) = \max_{i=1,..,K} \text{Score}(s_i, S)$
    return s^*
  end if
end procedure

3.2. Hyperparameter tuning and base learners selection

After introducing candidate machine learning models, hyperparameter tuning and base learners selection are conducted subsequently. These steps are shown in the left part of Figure 7, the ETA prediction framework.

In order to well select the hyperparameters and to achieve an unbiased performance of machine learning models, Nested Cross Validation (NCV) is proposed. It corresponds to the blue box on the left side of Figure 7. It consists of the outer loop and the inner loop. A K_1-fold cross validation splits the dataset S into K_1 subsets S_i, $i \in \{1, ..., K_1\}$. For each outer iteration i, $K_1 - 1$ folds $S_{-i} = S \backslash S_i$ act as training sets and one

fold S_i is test set. Then, there is another K_2-fold cross validation, which will further split the training sets S_{-i} into K_2 subsets $S_{-i,j}$, $j \in \{1, ..., K_2\}$. For each inner iteration j, $K_2 - 1$ folds $S_{-i} \backslash S_{-i,j}$ play the part of training sets and the remaining fold $S_{-i,j}$ is validation set. The purpose of the inner loop is to select hyperparameters, and the outer loop aims to assess the model's performance.

Taking $K_1 = 5$, $K_2 = 5$, the proportion of training sets, validation sets and test is set as 64%/16%/20%. Figure 8 illustrates the trajectory points of the training, validation, and test set of a possible cross validation fold in terms of QFUs. It can be seen that these trajectory points are between 20 and 25 Nm away from corresponding runways of BCIA. This fact proves that the spatial distribution of these sets is as close as possible, which helps to avoid training unbalanced models.

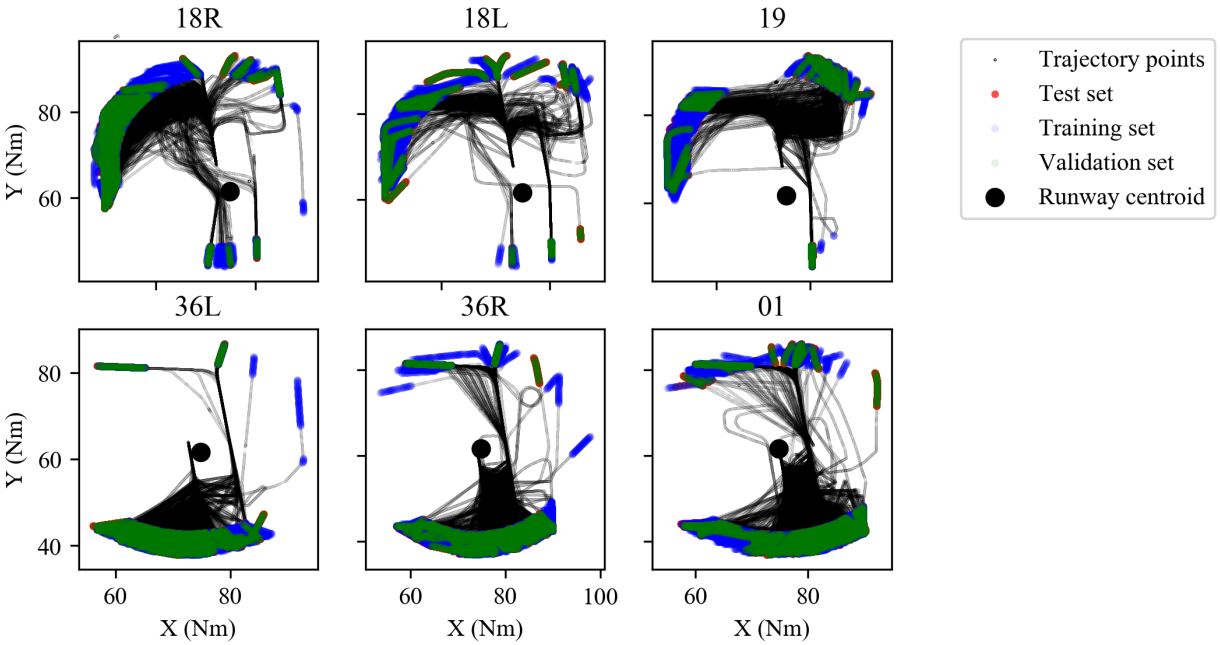


Figure 8: Trajectories in a possible fold of training, validation and test set in terms of QFUs

The random search algorithm is used for hyperparameter tuning in the inner CV. The idea of random search is to draw the hyperparameter independently from a probability distribution in the grid or range of hyperparameters. It is proved to be much more efficient than the classical grid search algorithm, especially in high-dimensional search space [27]. The random search is conducted 64 times for each model. The grids or ranges of hyperparameters and their probability distributions are presented in table 3. Other hyperparameters not in this table are set to default values.

The selection of base learners corresponds to the black box on the left side of Figure 7. To select the M_m base learners for stacking among N_m candidate models, some performance metrics are proposed to compare their prediction performances. The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are used as the error measures:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i| \tag{23}$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2}, \tag{24}$$

where $\hat{y}_i$ is the i-th predicted value of ETA and y_i is the i-th observed value of ETA.

Table 3: Hyperparameters optimized in random search

Method	Hyperparameter	Range or grid	Distribution
MLR	λ	$[10^{-5}, 10]$	log-uniform
FFNNs	M	$\{2, 3, ..., 10\} \cup \{16, 32, 64, 128\}$	uniform
	loss function	{'MAE', 'MSE'}	uniform
	Φ	{'ReLU', 'tanh', 'sigmoid'}	uniform
KNN	K	{1,2,...,20}	uniform
	weight function	{'uniform', 'inv_distance'}	uniform
	p [c]	{1,2,3,4,5,6}	uniform
GBM	boosting type	{'GBDT', 'GOSS', 'DART'}	uniform
	max number of leaves	{10, 20, ..., 100}	uniform
	fraction of bagging	{0.5, 0.7, 0.8, 0.9}	uniform
	fraction of feature	{0.5, 0.7, 0.8, 0.9}	uniform
	loss function	{'MAE', 'MSE'}	uniform
RF	number of estimators	{10, 20, ..., 100}	uniform
	max_features	[1, 2, 4, 8, 16, 32, 64, 128]	uniform
	loss function	{'MAE', 'MSE'}	uniform
ET		Same as RF	

[a] w kernel is 'poly'.
[b] Only used when kernel is 'rbf', 'poly' or 'sigmoid'.
[c] Only used when weight function is 'inv_distance'.

For each QFU, the cost function of i-th candidate model is defined as follows:

$$L(i) = \alpha \text{MAE}_i + \beta \text{RMSE}_i \tag{25}$$

where α and β are weights for different performance metrics. In this paper, MAE is considered to be as important as RMSE. Therefore, α and β are set to be equal. Then we select the M_m minimum value among $L(i), i = 1, ..., N_m$. These models are selected as the base learners for stacking in each QFU.

3.3. Model stacking

To further improve the prediction performance, the ensemble learning technique is proposed. Its main concept is to combine multiple models with a certain strategy to achieve better results than any individual model. More specifically, stacking [28] will be used in this study. Stacking can be regarded as a specific combination method that is combined by learning. Stacking first trains the first-level learners (individual learners) using the original dataset, and then generates a new dataset for training the second-level learner (combiner). In this new dataset, the output of the first-level learners is considered as input features, while the original output is still treated as output. In our case, individual learners are heterogeneous. The pseudocode of stacking is shown in Algorithm 4.

As illustrated in the right side of Figure 7, we use 2-layers stacking. In the first level of stacking with the i-th base learner, the dataset for stacking is firstly split into the training set (80%) and test set (20%). To avoid overfitting, the training set will be further split into new training sets and new test sets, similar to K-fold cross validation. The i-th base learner is trained from the new training set (in yellow) and makes predictions both on the new test sets (in red) and the initial test set (in blue). Then, the combined predicted values of new test sets are considered as training data, and the averaged predicted values of the initial test set are set to be test data. After concatenating i-th training and test data from $i = 1$ to M, we obtain the dataset for the second level of stacking. Conducting the same steps as the first layer, the dataset for the

Algorithm 4 Stacking

Input:
$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$: Training set
$\mathfrak{L}_{1,...,M}$: M Base learners
$\mathfrak{L}$: Meta learner
Output:
$H(\mathbf{x})$: Prediction output

procedure STACKING(S, $\mathfrak{L}_{1,...,M}$, $\mathfrak{L}$)
  for $m = 1$ to M **do**
    $h_m = \mathfrak{L}_m(D)$
  end for
  $D^* = \varnothing$
  for $n = 1$ to N **do**
    for $m = 1$ to M **do**
      $z_{nm} = h_m(\mathbf{x}_n)$
    end for
    $D^* = D^* \cup ((z_{n1}, z_{n2}..., z_{n_M}), y_n)$
  end for
  $h^* = \mathfrak{L}(D^*)$
  $H(\mathbf{x}) = h^*(h_1(\mathbf{x}), ..., h_M(\mathbf{x}))$
  return $H(\mathbf{x})$
end procedure

next level of stacking is generated. The final averaged output of the test data, which is the prediction of ETA, can be obtained by making use of the best base learners.

3.4. Performance evaluation

The performance evaluation of this study involves the evaluation of the effect of data preprocessing and stacking on ETA prediction. The performance metrics used are MAE and RMSE.

Firstly, The error measures of candidate machine learning models on the un-preprocessed dataset are computed. To evaluate the benefits of data preprocessing, for each prediction model, the overall performance metrics on the preprocessed dataset are computed using the prediction results of each QFU:

$$\text{MAE} = \sum_{i=1}^{Num} p_i \text{MAE}_i \tag{26}$$

$$\text{RMSE} = \sqrt{\sum_{i=1}^{Num} p_i (\text{RMSE}_i)^2} \tag{27}$$

where $Num = 6$, which is the total number of QFU. p_i is the proportion of flights for each QFU.

Next, the prediction performance of candidate machine learning models and the stacked model on the dataset for stacking will be evaluated and compared.

4. Experiment results and discussion

As mentioned before, the dataset for selecting base learners are used to evaluate the effect of data preprocessing and to select candidate machine learning models. The proportions of flights for each QFU are analyzed and presented in table 4.

Table 4: The proportions of flights for each QFU in the dataset for selecting base learners and for stacking

QFU	Percentage	
	For selecting base learners	For stacking
18R	21.32%	24.44%
18L	4.21%	3.23%
19	9.51%	5.40%
36L	23.72%	28.69%
36R	15.09%	13.03%
01	26.14%	25.21%

Table 5: ETA prediction results in terms of each QFU on the preprocessed dataset for selecting base learners

Models	QFU-18R		QFU-18L		QFU-19	
	MAE (s)	RMSE (s)	MAE (s)	RMSE (s)	MAE (s)	RMSE (s)
MLR	**48.81**	**73.86**	75.68	125.84	80.44	104.47
FFNNs	**44.90**	**71.74**	**74.69**	**135.84**	**76.91**	**110.00**
GBM	**44.02**	**67.40**	**63.68**	**101.69**	**75.35**	**99.83**
KNN	58.92	85.35	75.90	109.53	83.92	110.36
RF	50.21	78.03	77.06	147.84	78.26	104.75
ET	49.45	76.34	**73.71**	**135.82**	**73.06**	**97.37**

Models	QFU-36L		QFU-36R		QFU-01	
	MAE (s)	RMSE (s)	MAE (s)	RMSE (s)	MAE (s)	RMSE (s)
MLR	47.86	64.62	68.30	97.66	85.45	154.18
FFNNs	**35.67**	**55.12**	**54.29**	**86.08**	**58.01**	**122.19**
GBM	**36.90**	**52.86**	**58.97**	**97.30**	**64.38**	**110.16**
KNN	51.53	72.64	75.55	116.57	86.34	139..05
RF	38.99	60.90	61.34	101.52	68.05	161.25
ET	**38.26**	**58.60**	**59.48**	**96.36**	**67.04**	**157.11**

We then evaluate the ETA prediction performance of 6 proposed candidate machine learning models on the preprocessed and un-preprocessed dataset.

The prediction results of candidate machine learning models for 6 QFUs are summarized in table 5. The best MAE and RMSE value of each QFU is bolded. The second and third best performance metrics are bolded and colored in gray. Among these candidate models, FFNNs are the best model in 3 QFUs, GBM has the best performance in 2 QFUs, and ET performs best in only one of the QFU. KNN performs the worst. In addition, for all QFUs, FFNNs and GBM are in the top 3 best models. MLR occasionally has good performance. According to the loss function, FFNNs, GBM, and ET are selected as the base learners for QFU-18L, QFU-19, QFU-36L, QFU-36R, and QFU-01. MLR, FFNNs, and GBM are selected as the base learners for QFU-18R. Note that, during the nested cross validation, hyperparameters of these models are optimized and are utilized in the stacked model.

The error metrics of the candidate models on the preprocessed and un-preprocessed dataset are shown in Table 6. After preprocessing the dataset, the predictions of all candidate models become much more accurate, especially for MLR. It can be seen that the prediction of MLR is not satisfying on the raw dataset. However, The preprocessing step greatly improves its prediction accuracy. Though MLR is a linear model, Its predictive performance is even close to other complex machine learning models in the un-preprocessed dataset. This fact proves that data preprocessing proposed in this study could greatly improve the quality

Table 6: Overall ETA prediction results on the preprocessed and un-preprocessed dataset for selecting base learners

Models	Preprocessed dataset		Un-preprocessed dataset	
	MAE (s)	RMSE (s)	MAE(s)	RMSE (s)
MLR	65.25	107.30	88.96	131.15
FFNNs	51.86	93.64	57.61	108.41
GBM	53.71	87.27	55.85	90.88
KNN	69.94	107.58	78.38	125.55
RF	57.69	111.78	60.70	119.97
ET	56.18	107.66	58.27	109.53

of raw data and boost the prediction accuracy. It is also robust to outliers, missing data, and noise.

Then, the overall ETA prediction performance is analyzed on the dataset for stacking. Similar to the previous steps, the prediction results in terms of each QFU are summarized, and the overall prediction results are calculated according to the proportions of flights for each QFU, depicted in table 4. In this way, the prediction results of all candidate machine learning models and the stacked model are summarized in Table 7. The minimum error metrics for each QFU is bolded. Thanks to data preprocessing, the ETA prediction results of all models seem satisfying. More specifically, it can be observed that the effect of stacking in this prediction task is significantly positive. In comparison with other candidate models, the stacked model has the minimum prediction error for all QFUs, except for QFU-01. Though the performance improvement of a few QFUs is not obvious, for most QFUs, the stacked model reduces the MAE by nearly 2 seconds and the RMSE by over 3 seconds. This fact reveals that stacking is more capable of extracting hidden information in the ETA prediction task.

Table 7: ETA prediction results of candidate machine learning models and stacked model on the dataset for stacking

Models	QFU-18R		QFU-18L		QFU-19	
	MAE (s)	RMSE (s)	MAE (s)	RMSE (s)	MAE (s)	RMSE (s)
MLR	49.77	68.72	65.15	91.69	74.63	100.17
FFNNs	44.35	62.86	51.82	71.40	70.75	97.54
GBM	44.92	63.86	74.74	99.48	68.46	89.14
KNN	53.98	75.52	109.02	290.13	80.53	106.36
RF	47.87	68.51	95.39	223.08	75.43	98.57
ET	47.60	67.21	88.61	183.57	73.82	96.54
Stacked model	**44.26**	**62.84**	**49.62**	**67.18**	**66.25**	**87.98**

Models	QFU-36L		QFU-36R		QFU-01		Total	
	MAE (s)	RMSE (s)	MAE (s)	RMSE (s)	MAE (s)	RMSE (s)	MAE (s)	RMSE (s)
MLR	47.27	64.64	69.92	112.94	79.65	126.31	61.03	94.07
FFNNs	36.06	56.72	51.83	109.38	67.04	124.20	50.32	89.52
GBM	36.29	57.82	56.22	118.12	61.79	**113.02**	50.38	87.98
KNN	47.55	70.71	70.21	146.46	73.99	125.91	62.48	113.79
RF	39.51	61.69	62.76	135.67	68.88	120.09	55.71	102.00
ET	39.17	60.42	57.29	116.68	66.11	118.00	53.83	95.00
Stacked model	**35.31**	**55.45**	**48.45**	**104.97**	**61.23**	113.42	**47.86**	**84.19**

5. Conclusion

In this paper, a novel automated data-driven ETA prediction framework is presented, implemented, and experimented. Based on open historical ADS-B data of BCIA, the proposed framework mainly consists of data preprocessing and machine learning prediction models. The data preprocessing is capable of processing dense 4D trajectory data of all arrival flights in TMA and handling complicated traffic patterns. It even can be extended to other TMAs without manually tuning the hyperparameters. The experiments were firstly performed on the dataset for selecting base learners, using candidate machine learning models with nested cross validation. A comparative study on the preprocessed and un-preprocessed dataset is conducted and proves that the proposed preprocessing steps are robust in dealing with outliers, missing points, and noises. It could greatly improve the accuracy of ETA prediction. The base learners for each QFU are selected and their hyperparameters are determined. Furthermore, an ensemble learning strategy stacking is introduced. The concept of stacking is to combine well-performed individual models with certain strategies. The result proves that the stacked model outperforms other individual models both in terms of accuracy and stability.

In future work, we will extend our approach to other TMAs, and make ETA prediction on larger datasets with more special scenarios. In addition, the prediction of ETA on irregular flights should be studied as well.

6. Acknowledgement

The authors would like to thank Serge Roux for providing the ADS-B data of BCIA.

References

[1] Z. Wang, M. Liang, and D. Delahaye, "Automated data-driven prediction on aircraft estimated time of arrival," in *Sesar Innovations Days 2018*, 2018.

[2] G. Enea and M. Porretta, "A comparison of 4D-trajectory operations envisioned for nextgen and sesar, some preliminary findings," in *28th Congress of the International Council of the Aeronautical Sciences*, 2012, pp. 23–28.

[3] Dart – data-driven aircraft trajectory prediction research. [Online]. Available: http://dart-research.eu/

[4] M. Ghasemi Hamed, D. Gianazza, M. Serrurier, and N. Durand, "Statistical prediction of aircraft trajectory: regression methods vs point-mass model." ATM Seminar, 2013.

[5] W. Kun and P. Wei, "A 4-d trajectory prediction model based on radar data," in *Control Conference, 2008. CCC 2008. 27th Chinese*. IEEE, 2008, pp. 591–594.

[6] A. de Leege, M. van Paassen, and M. Mulder, "A machine learning approach to trajectory prediction," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013, p. 4782.

[7] S. Hong and K. Lee, "Trajectory prediction for vectored area navigation arrivals," *Journal of Aerospace Information Systems*, vol. 12, no. 7, pp. 490–502, 2015.

[8] K. Tastambekov, S. Puechmorel, D. Delahaye, and C. Rabut, "Aircraft trajectory forecasting using local functional regression in sobolev space," *Transportation research part C: emerging technologies*, vol. 39, pp. 1–22, 2014.

[9] Y. Le Fablec and J.-M. Alliot, "Using neural networks to predict aircraft trajectories." in *IC-AI*, 1999, pp. 524–529.

[10] R. Alligier, D. Gianazza, and N. Durand, "Machine learning applied to airspeed prediction during climb," in *ATM seminar 2015, 11th USA/EUROPE Air Traffic Management R&D Seminar*, 2015.

[11] ——, "Machine learning and mass estimation methods for ground-based aircraft climb prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3138–3149, 2015.

[12] Z. Wang, M. Liang, and D. Delahaye, "Short-term 4d trajectory prediction using machine learning methods," in *SID 2017, 7th SESAR Innovation Days*, 2017.

[13] M. G. Karlaftis and E. I. Vlahogianni, "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 3, pp. 387–399, 2011.

[14] CAAC Operation Monitoring Center, "Report on national civil aviation flight operation efficiency in 2017 (chinese)," Tech. Rep., 2018.

[15] C. Aeronautical Information Center, ATMB, "Electronic aeronautical information publication of people's republic of china," Tech. Rep., 2017.

[16] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] F. Chollet *et al.*, "Keras: Deep learning library for theano and tensorflow," *URL: https://keras. io/k*, vol. 7, no. 8, p. T1, 2015.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[20] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.
[21] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.
[22] K. Rashmi and R. Gilad-Bachrach, "Dart: Dropouts meet multiple additive regression trees," in *International Conference on Artificial Intelligence and Statistics*, 2015, pp. 489–497.
[23] 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.
[24] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[25] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, NY, USA:, 2001, vol. 1, no. 10.
[26] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.
[27] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.
[28] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.