Data-driven Conflict Detection Enhancement in 3D Airspace with Machine Learning

Zhengyi Wang, Man Liang, Daniel Delahaye

Data-driven Conflict Detection Enhancement in 3D Airspace with Machine Learning

Zhengyi WANG *
Ecole Nationale
de l'Aviation Civile
Toulouse, France
zhengyi.wang@enac.fr

Man LIANG
University of
South Australia
Adelaide, Australia
annie.liang@unisa.edu.au

Daniel DELAHAYE
Ecole Nationale
de l'Aviation Civile
Toulouse, France
delahaye@recherche.enac.fr

***Abstract*—Trajectory prediction with Closest Point of Approach (CPA) concept is a fundamental element of aircraft Conflict Detection (CD) problem. Conventional motion-based CPA prediction model generally assumes that aircraft is flying in straight line with constant speed. But due to environment uncertainties and ground speed changes, this conventional method frequently lacks accuracy in the real world with a high rate of false alarms and missed detections. In this paper, we introduce a novel automated data-driven CD framework with Machine Learning (ML) for 3D CPA prediction in a lookahead time of less than 20 minutes. Firstly, a 3D CPA model with cylindrical norm is proposed as the baseline. Then, data preparation with Mode-S observation data in France is explained, including data collection and data processing, to convert raw Mode-S data to the close-to-reality dataset. Furthermore, feature engineering is applied to build up a feature set with 16 features. Finally, four prevailing ML models are used to predict the time, horizontal distance and vertical distance of CPA in 3D airspace. CD is conducted based on the predicted values. The prediction and CD results show that all proposed ML models outperform the baseline model. Especially, GBM and FFNNs could strongly enhance the performance of CD.**

***Keywords*—Air Traffic Management, 3D Conflict Detection, Closest Point of Approach, Machine Learning**

1. INTRODUCTION

Safety is the primary concern in aviation due to the low acceptance of risk. It is typically quantified in terms of conflicts. A conflict is described as a loss of separation between two or more aircraft. In 3D airspace, the separation requirement for two aircraft involves minimum horizontal separation, which is typically 5Nm in en-route airspace under radar surveillance, and minimum vertical separation, which is 1000ft below FL290 and 2000ft above FL290 in Non-Reduced Vertical Separation Minimum (Non-RVSM) airspace [1]. The prevention of conflicts consists of conflict detection and resolution (CD&R). In CD stage, future positions of aircraft during specific lookahead time interval are computed, named Trajectory Prediction (TP). A potential conflict is declared if at any of these timestamps the aircraft are in loss of separation. In conflict resolution case, the aircraft trajectories involved in the conflict are re-planned to avoid the conflict [2]. CD is generally studied at three different levels: long-term, mid-term and short-term [2]. In long-term, CD involves trajectory planning and airline scheduling, which are the first operations to avoid unnecessary conflicts and to ensure flight safety. Mid-Term Conflict Detection (MTCD) is usually carried out by Air Traffic Controllers (ATCOs) with the supports of semi-automated tools over a time horizon of tens of minutes. The frequently used tools include Center TRACON Automation System (CTAS) [3] and User Request Evaluation Tool (URET) [4], etc. In view of Short-Term Conflict Detection (STCD), the time scale is in seconds or minutes. The detected conflict must be dealt with immediately, otherwise it will cause a severe accident. To assist ATCOs and pilots, Short Term Conflict Alert (STCA) and Collision Avoidance System (TCAS) [5] are developed and applied. The accuracy of TP plays a vital role in the CD&R and air safety. In this paper, automated STCD and MTCD with the lookahead time less than 20 minutes will be studied through a data-driven CD framework based on Machine Learning (ML) models.

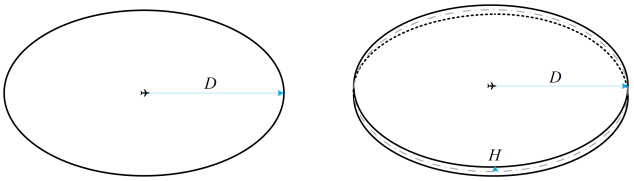
Closest Point of Approach (CPA) is a key concept in the algorithmic level for STCD and MTCD. It is intended to determine the minimum distance and the associated time between two aircraft at the same altitude with crossing or converging traffic. The CPA is firstly introduced in maritime domain for vessel CD problem [6, 7]. Then, plenty of researches have been carried out on using CPA concept for 2D aircraft CD problem [2, 8–14]. CPA is also extended to be applied in 3D airspace [15, 16]. In practical application for Air Traffic Management (ATM), the CPA concept was successfully applied in Eurocontrol's MTCD tool, FAA AERA-2 tool, URET [13].

However, CPA could be problematic. In theoretical CPA calculation, it is assumed that the aircraft flys in a straight trajectory with a constant velocity vector. Actually, the aircraft may change or intend to change the heading and vertical rate throughout the flight. Even in cruise phase, there are still some minor changes in heading. Besides, the ground speed is the superposition of airspeed and wind speed vectors. These two vector components both have high-level uncertainties, especially wind speed. Owing to the stochastic climate change and the limitation of wind modelling strategy, the wind prediction bias is large and increases with time. Therefore, the traditional motion-based CPA calculation method frequently lacks accuracy in the real world with a high rate of false alarms and missed detections [13]. Instead, data-driven techniques

are able to train appropriate models from all relevant and actual past data with no or few prior assumptions and few requirements for data quality. Compared to classical model-driven approaches, contextual features can be extracted, such as aircraft intents, speed changes, wind effects, etc.

Machine learning is one of the most prevailing techniques in data-driven approaches. Typical machine learning models for 4D TP include linear regression models and Neural Networks (NNs) models [17]. In terms of 2D CD, preliminary efforts have been done by the authors and the validity of ML approaches was proved [18]. Thus the objective of this paper is to extend the 2D CD to 3D CD with ML methods. In the experiments, we will firstly create a close-to-reality dataset from Mode-S observations. The generated dataset ensures that all aircraft are flying freely with no further conflict resolution maneuvers. Then, feature set is generated based on the nature of this problem. Prevailing ML methods are built to predict the time, horizontal distance and vertical distance of CPA. These models will be ranked based on their performance. To the best our knowledge, Most CD researches use non-actual data. This study will provide a novel insight of 3D CD based on actual trajectory data.

2. 3D Closest Point of Approach Model



(a) 2D (b) 3D

Figure 1: Protected zone

In broad terms, CPA refers to the positions at which two dynamically moving objects reach their closest possible distance, in case of no change in heading, rate of climb and speed [19]. The distance and time at the CPA are crucial in determining if and when the aircraft are in loss of separation. In view of 2 aircraft travelling in the same horizontal plane, the protected zone is a circle, as shown in Figure 1a. Their minimum Euclidean distance is reached at CPA, which is discussed in detail in our previous research [18].

However, in 3D airspace, the protected zone becomes a cylinder with 30 or 60 times wider than high, as illustrated in Figure 1b. The Euclidean distance is no longer a proper metric to determine the CPA between aircraft. For example, a loss of separation may not occur at the time that minimizes their 3D Euclidean distance. To this end, a distance metric called cylindrical norm is proposed and proved to be effective [15]:

Definition 1: Given a cylinder of radius D and half-height H, the cylindrical norm of a vector $\mathbf{w} \in \mathbb{R}^3$ is defined as follows:

$$\|\mathbf{w}\|_{\text{cyl}} = \max\left(\frac{\|\mathbf{w}_z\|}{H}, \frac{\|\mathbf{w}_{(x,y)}\|}{D}\right) \quad (1)$$

We focus on 2 aircraft (ownship, intruder) flying in 3D airspace over a lookahead time interval from T_l to T_u. It is assumed that each aircraft is represented by a point flying at constant speed along a linear trajectory. As shown in Figure 2, their velocity vectors are $\mathbf{v}_o$ and $\mathbf{v}_i$. their position vectors at time $t \in [T_l, T_u]$ are $\mathbf{p}_{o,t}$ and $\mathbf{p}_{i,t}$. Let the relative position vector be $\mathbf{p}_t = \mathbf{p}_{o,t} - \mathbf{p}_{i,t}$ and the relative velocity vector be $\mathbf{v} = \mathbf{v}_o - \mathbf{v}_i$. The property of time of CPA is provided in the following definition:

Definition 2: The time of CPA $t_{\text{CPA}} \in [T_l, T_u]$ satisfies

$$\forall t \in [T_l, T_u],\ \|\mathbf{s}_t + t\mathbf{v}\|_{\text{cyl}} \geq \|\mathbf{s}_{t_{\text{CPA}}} + t_{\text{CPA}}\mathbf{v}\|_{\text{cyl}} \quad (2)$$

Thus, the time of CPA can be determined as follows:

$$t_{\text{CPA}} = \arg\min_{t \in [T_l, T_u]} \|\mathbf{s}_t + t\mathbf{v}\|_{\text{cyl}} \quad (3)$$

To solve (3), algorithm 1 is proposed [19]. The main idea of this algorithm is to minimize the square of $\|\mathbf{s}_t + t\mathbf{v}\|_{\text{cyl}}$, and study the specific values of two quadratic polynomials functions.

After calculating t_{CPA}, according to the following theorem, it can be judged whether 2 aircraft are in loss of separation:

Theorem 1: Aircraft are in conflict if and only if $\|\mathbf{s}_{t_{\text{CPA}}} + t_{\text{CPA}}\mathbf{v}\|_{\text{cyl}} < 1$.

In this study, we will perform STCD and MTCD on aircraft pairs, in which the t_{CPA} is within the lookahead interval less than 20 mins. This method is referred to as fixed threshold CD, which is used in Eurocontrol's MTCD tool [13]. Remark that, we refer this conventional CD approach as the CPA baseline method.

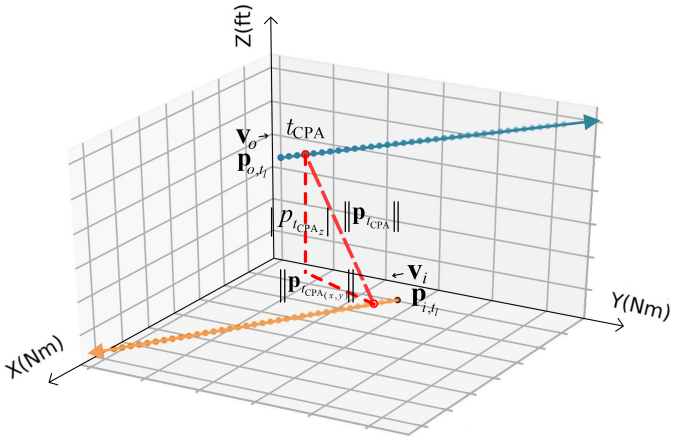


Figure 2: illustration of CPA between 2 aircraft in 3D airspace

3. Dataset Preparation with Mode-S Data

A. Data collection

Mode-S data covering France on January 20, 2012 is used in this study. It contains 11,214,216 data records of 21,314 trajectories. Each trajectory data record consists of the following information reported by aircraft: flight number, Coordinated

Algorithm 1 Time of CPA calculation in 3D airspace

Input:

$\mathbf{s}, \mathbf{v}$: Relative position and velocity vector of 2 aircraft at the initial state

T_l, T_u: lower bound and upper bound of lookahead time interval

D, H: Horizontal and vertical separation

Output:

t_{CPA}: Time of Closest Point of Approach

procedure TCA($\mathbf{s}$, $\mathbf{v}$, T_l, T_u, H, D)
  $A \leftarrow v_z^2/H^2 - \|\mathbf{v}_{(x,y)}\|^2/D^2$
  $B \leftarrow 2s_z v_z/H^2 - 2(\mathbf{s}_{(x,y)} \cdot \mathbf{v}_{(x,y)})/D^2$
  $C \leftarrow s_z^2/H^2 - \|\mathbf{s}_{(x,y)}\|^2/D^2$
  $T \leftarrow \varnothing$
  if $\mathbf{v}_{(x,y)} \neq \mathbf{0}$ **then**
    $T \leftarrow T \cup \left\{-(\mathbf{s}_{(x,y)} \cdot \mathbf{v}_{(x,y)})/\|\mathbf{v}_{(x,y)}\|^2\right\}$
  end if
  if $v_z \neq 0$ **then**
    $T \leftarrow T \cup \{-s_z/v_z\}$
  end if
  if $A \neq 0$ and $B^2 - 4AC \geq 0$ **then**
    for $\alpha \in \{-1, 1\}$ **do**
      $T \leftarrow T \cup \left\{(-B + \alpha\sqrt{B^2 - 4AC})/(2A)\right\}$
    end for
  else if $A = 0$ and $B \neq 0$ **then**
    $T \leftarrow T \cup \{-C/B\}$
  end if
  $\tau \leftarrow T_l$
  for $t \in T$ **do**
    if $t > T_l$ and $\|\mathbf{s} + t\mathbf{v}\|_{\mathrm{cyl}} < \|\mathbf{s} + \tau\mathbf{v}\|_{\mathrm{cyl}}$ **then**
      $\tau \leftarrow t$
    end if
  end for
  $t_{\mathrm{CPA}} \leftarrow \min(\tau, T_u)$
  return t_{CPA}
end procedure

Universal Time (UTC) timestamp, Position (longitude, latitude, altitude), ground speed g, vertical speed v, heading φ, wind direction δ and wind speed w. Note that data records with the same flight number belong to the same trajectory.

As a matter of fact, conflicts occur rarely in the real world. All potential conflicts have generally been resolved by ATCOs, pilots or automated systems. As a result, there is no conflict in the raw dataset. To create more potential conflict cases close to the actual situation, we choose to align the initial timestamps of all trajectories to 0. Each trajectory pair is then designed to ensure that aircraft are flying freely at their speed and heading without conflict resolution maneuvers. The actual aircraft intent is still inside each trajectory. After time alignment, it is found that the traffic situation contains much more potential conflicts and becomes more complicated, which highly increases the difficulty to realize efficient and accurate CDs.

Finally, the records of all trajectories are stored in $N = 21,314$ matrices $\mathbf{T}^{(i)}, i = 1, ..., N$. Each matrix $\mathbf{T}^{(i)}$ is of size $L_i \times M$, where L_i is the length of i-th trajectory and $M = 9$ is the number of attributes except flight number.

B. Data processing

To convert raw Mode-S data to the close-to-reality dataset, the following data processing stages are applied:

1) Coordinate system transformation: It is necessary to convert the Geographic Coordinate System (GCS) into Projection Coordinate System (PCS) in that calculating distance using longitude and latitude is not straightforward. More specifically, the spatial reference EPSG 2154 is selected as the PCS. The area of use for this PCS is France. The projection method is Lambert conformal Conic (LCC), which is widely used in aeronautical charts. After coordinate system transformation, longitude, latitude and altitude are converted into X, Y, Z coordinates.

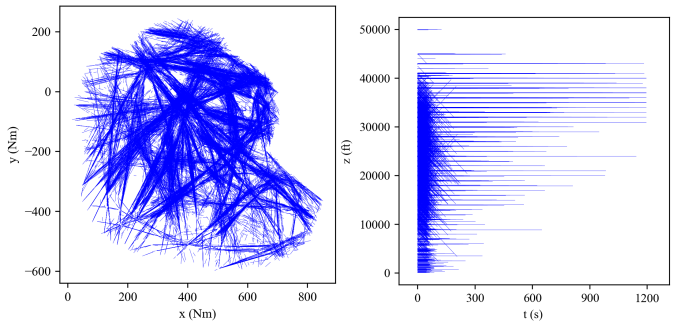
2) Missing point estimation: Normally, the mode-S data updates every 4 seconds. However, in some cases the update frequency in the raw dataset can be 8, 16, 32 seconds. in this study, these missing data records are estimated by using piecewise linear interpolation. Note that, interpolate latitude and longitude values will bring error to the calculation of distance, because the length of the meridian at different latitude is different. Thus, the interpolation will be performed after transforming the coordinate system.

3) Cyclical attribute conversion: The heading in raw data is magnetic heading, which is in relation to magnetic north. Compared with the heading estimated by position changes in the current coordinate system, the error is negligible. Therefore, the heading doesn't need to be recalculated. It is noteworthy that heading and wind direction are cyclical attributes. If we directly calculate the difference between 2 angles, for example, $1^\circ - 0^\circ$ and $359^\circ - 0^\circ$ get completely different results. However, the results should be 1 degree in both cases. To this end, we create new attributes, deriving a sine and a cosine transform. Sine and cosine functions are both uniformly continuous on $\mathbb{R}$, which makes the heading and wind direction cyclical. Then, the difference can be obtained by calculating the Euclidean distance between sine and cosine values.

4) Trajectory segmentation: To increase the robustness of the proposed CPA calculation method, we allow fluctuations of speed and heading in the dataset. A sensitivity analysis on 2D CPA [12] has found that the fluctuation of heading has a much greater impact on the CPA calculation than the fluctuation of speed. The fluctuation of speed will not result in the deviation from the original route, but the fluctuation of heading will deviate the aircraft from original trajectory and make it unpredictable. Thus, a trajectory will be filtered if the fluctuation of heading is greater than 2 degrees or the rate of climb changes. To improve the data quality, we propose a trajectory segmentation method. For every trajectory, the distance of heading and the difference of vertical speed between all adjacent points are calculated. Then, points that

do not meet the criterion are considered as split points. These points will split the trajectory to smaller trajectory segments, which must contain at least 20 points.

5) Trajectories matching: Trajectories matching aims to match trajectories into pairs and detect if there are conflicts. If all N aircraft are involved, $N(N-1)/2$ pairs will be matched. Considering that it is a large value and aircraft that are far apart should not be involved, before calculating the minimum cylindrical distances between every points in each trajectory pair, we simply calculate the minimum vertical $v_{\min}$ and horizontal distance $h_{\min}$ between 2 start points and end points of each trajectory pair. If $v_{\min}/H > v_{\text{threshold}}$ or $h_{\min}/D > h_{\text{threshold}}$, this trajectory pair is filtered. $v_{\text{threshold}}$ is set to 2 and $h_{\text{threshold}}$ is set to 3 according to the experience. For other trajectory pairs, the minimum cylindrical distances are calculated. The minimum duration of 2 trajectories is set as the lookahead time for each trajectory pair. Finally, the remaining trajectories in the dataset are shown in Figure 3. Unlike data preprocessing in the previous work [18], It can be seen that most trajectories in the raw data are kept. Figure 3a shows trajectories projected in the horizontal plane. It can be seen that the headings of aircraft are almost constant. Figure 3b presents the altitudes of trajectories that change over time. The rate of climb of each trajectory segment includes both cases: descent, climb and level flight.



(a) Projection in the horizontal plane (b) Altitudes change over time

Figure 3: Illustration of trajectories in the dataset

4. Machine Learning Approaches to Predict Closest Point of Approach and Detect Conflict

A. Feature engineering

Feature engineering refers to the process of extracting features from raw data using domain knowledge and transforming them into formats which are suitable for ML models.

The feature set in this study should be built based on the nature of CPA and the dataset. To begin with, the heading and the rate of climb of aircraft in the dataset are nearly constant, which roughly conforms to the assumptions of the conventional CPA calculation. However, the speed changes greatly. The initial speed cannot reflect the aircraft intent in the lookahead time. To this end, information on multiple trajectory points should be used to construct features. In this paper, we focus on the first 5 trajectory points of each trajectory pair, including the attributes of ownship ($X_{o,t}$, $Y_{o,t}$, $Z_{o,t}$, $\varphi_{o,t}$, $g_{o,t}$, $v_{o,t}$, $\delta_{o,t}$, $w_{o,t}$) and of intruder ($X_{i,t}$, $Y_{i,t}$, $Z_{i,t}$, $\varphi_{i,t}$, $g_{i,t}$, $v_{i,t}$, $\delta_{i,t}$, $w_{i,t}$), $t = 1, ..., 5$. The lookahead time T is set as the elapsing time from the 5th trajectory point to the last trajectory point and is included in the feature set.

- $T = T_f - T_5$

However, some attributes are not related to the CPA calculation. For example, the CPA is derived based on the current position, regardless of the previous positions. Thus, the relative position of the 5th trajectory points is added into the feature set:

- $X_5 = X_{o,5} - X_{i,5}$
- $Y_5 = Y_{o,5} - Y_{i,5}$
- $Z_5 = Z_{o,5} - Z_{i,5}$

In addition, the values of some attributes on the first 5 trajectory points may be close, including heading, ground speed, wind direction and wind speed. Simply setting the first 5 attributes as lag features may bring redundancy to the feature set. The prediction will be much less effective if the inputs have many similar values. Since the values of heading, vertical speed and wind direction rarely change in the first 5 trajectory points of aircraft, we keep the value of attributes that appears the most often:

$$\varphi_o = \text{mode}(\varphi_{o,1}, ..., \varphi_{o,5}) \tag{4}$$
$$\varphi_i = \text{mode}(\varphi_{i,1}, ..., \varphi_{i,5}) \tag{5}$$
$$\delta_o = \text{mode}(\delta_{o,1}, ..., \delta_{o,5}) \tag{6}$$
$$\delta_i = \text{mode}(\delta_{i,1}, ..., \delta_{i,5}) \tag{7}$$
$$v_o = \text{mode}(v_{o,1}, ..., v_{o,5}) \tag{8}$$
$$v_i = \text{mode}(v_{i,1}, ..., v_{i,5}) \tag{9}$$

where mode function returns the most frequently occurring number.

The cosine and sine value of heading and wind direction are created as features:

- $\cos\varphi_o$, $\sin\varphi_o$
- $\cos\varphi_i$, $\sin\varphi_i$
- $\cos\delta_o$, $\sin\delta_o$
- $\cos\delta_i$, $\sin\delta_i$

The difference between 2 vertical speeds is added to the feature set:

- $v = v_i - v_o$

Inspired by the use of convolution for denoising in signal processing, we define a new "convolution" operation called average subtractive convolution to process ground speed and wind speed.

Definition 3: The average subtractive convolution of 2 vectors of equal length $\mathbf{a} = (a_1, a_2, ..., a_n)$ and $\mathbf{b} = (b_1, b_2, .., b_n)$ is given by:

$$\mathbf{a} * \mathbf{b} = \frac{1}{n^2} \sum_{i=1}^{2n-1} \sum_{j=\max(1,i-n+1)}^{\min(i,n)} (b_{i-j+1} - a_j) \tag{10}$$

Therefore, the following features involved with ground speed vectors $\mathbf{g}_i = (g_{i,1}, ..., g_{i,5})$, $\mathbf{g}_o = (g_{o,1}, ..., g_{o,5})$ and wind speed vectors $\mathbf{w}_i = (w_{i,1}, ..., w_{i,5})$, $\mathbf{w}_o = (w_{o,1}, ..., w_{o,5})$ are added to the feature set:

- $g_x = (\cos\varphi_i \cdot \mathbf{g}_i) * (\cos\varphi_o \cdot \mathbf{g}_o)$
- $g_y = (\sin\varphi_i \cdot \mathbf{g}_i) * (\sin\varphi_o \cdot \mathbf{g}_o)$
- $w_x = (\cos\delta_i \cdot \mathbf{w}_i) * (\cos\delta_o \cdot \mathbf{w}_o)$
- $w_y = (\sin\delta_i \cdot \mathbf{w}_i) * (\sin\delta_o \cdot \mathbf{w}_o)$

In addition, some additional features derived from conventional CPA calculation algorithm are also introduced:

- $A = v^2/H^2 - \|\mathbf{g}\|^2/D^2$
- $B = 2s_{5_z}v/H^2 - 2(\mathbf{s}_{5_{(x,y)}} \cdot \mathbf{g})/D^2$
- $C = s_{5_z}^2/H^2 - \|\mathbf{s}_{5_{(x,y)}}\|^2/D^2$

where $\mathbf{g} = (g_x, g_y)$, $\mathbf{s}_5 = (X_5, Y_5, Z_5)$.

Finally, the feature set contains 16 features. All useful information for calculating CPA is included in the feature set.

B. Conflict detection model

Several ML approaches are applied to predict the time of CPA t_{CPA}, horizontal distance of CPA $d_{\mathrm{CPA}_{xy}}$ and vertical distance of CPA d_{CPA_z}, based on real trajectory dataset generated in section 3. Note that, for each ML algorithm, 3 single-target models will be built. Each model is trained on the training set $\mathcal{S}_i : (\mathbf{X}, \mathbf{t}^{(i)}) = \{(\mathbf{x}_1, t_1^{(i)}), ..., (\mathbf{x}_N, t_N^{(i)})\}, i = 1, 2, 3$. After computation, the predicted cylindrical norm of relative position at CPA is calculated as:

$$\|\mathbf{d}_{\mathrm{CPA}}\|_{\mathrm{cyl}} = \max\left(\frac{d_{\mathrm{CPA}_z}}{H}, \frac{d_{\mathrm{CPA}_{xy}}}{D}\right) \tag{11}$$

Then, the following formula is used to detect conflict:

$$\begin{cases} \|\mathbf{d}_{\mathrm{CPA}}\|_{\mathrm{cyl}} < 1, & \text{Conflict} \\ \|\mathbf{d}_{\mathrm{CPA}}\|_{\mathrm{cyl}} \geq 1, & \text{No conflict} \end{cases} \tag{12}$$

C. Machine learning models

In our previous research [18], the results demonstrate that Multiple Linear Regression (MLR) and Support Vector Machine (SVM) is not suitable to address our problem. Thus these models are eliminated. We propose other prevailing ML models, including Feed-Forward Neural Networks (FFNNs), K-Nearest Neighbors (KNN), Gradient Boosting Machine (GBM), and Random Forests (RF). Details of these models are described in the appendix of this paper.

5. Model Validation and Results Discussion

The experiment was run on a laptop with Intel core i7-8750H CPU @ 2.20GHz, 16GB RAM and NVIDIA GeForce GTX 1070 GPU. All algorithms were implemented in Python 3.6.2. Note that the dataset is large and the models are complex. The training process could be time-consuming. Nonetheless, the training is completed offline. Then, the real-time prediction can be fully ensured.

A. Model selection

In order to well select the hyperparameters and to achieve an unbiased performance of ML models, the Nested Cross Validation (NCV) is proposed. It consists of outer loops and inner loops. A K_1-fold CV splits the dataset S into K_1 subsets S_i, $i = 1, ..., K_1$. For each outer loop i, S_i is the test set and the remaining $K_1 - 1$ folds $S_{-i} = S \backslash S_i$ act as the training set. Then, there is another K_2-fold CV, which will further split the training sets S_{-i} into K_2 subsets $S_{-i,j}$, $j = 1, ..., K_2$. For each inner loop j, $S_{-i,j}$ act as the validation set and the remaining $K_2 - 1$ folds $S_{-i} \backslash S_{-i,j}$ act as the training set. The purpose of the inner loop is to select the hyperparameters and the outer loop aims to assess the model performance. Let $K_1 = 5$, $K_2 = 5$, then the proportion of training sets, validation sets and test is respectively $64\%/16\%/20\%$.

Random search is used for hyperparameter optimization in the inner CV. It is much more efficient than grid search algorithm, especially when the search space has high dimension. The pseudocode of random search is depicted in Algorithm 2. The strategy of random search is to draw independently from a probability distribution in the grid or range of hyperparameters.

Algorithm 2 Random search on model $\mathcal{A}$

Input:
$\mathbf{V} = \{\mathbf{v}_1, ..., \mathbf{v}_N\}$: Range or grid of N hyperparameters
$\mathbf{F} = \{F_1(\cdot), ..., F_N(\cdot)\}$: Cumulative Distribution Function (CDF) of N hyperparameter values
S_V: Validation set
K: Numbers of sampling iterations
Output:
λ^*: Selected hyperparameters

procedure RandomSearchCV($\mathbf{V}$, $\mathbf{F}$, S_V, K)
  for i = 1 to K **do**
    $(r_1, .., r_N) \leftarrow N$ uniformly distributed random numbers generated between 0 and 1
    $\Lambda_i \leftarrow (F_1^{-1}(r_1), ..., F_N^{-1}(r_N))$ ▷ Inverse CDF
  end for
  $\lambda^* \leftarrow \arg\min_{\lambda \in \Lambda} Err(\mathcal{A}, \lambda, S_V)$
  return λ^*
end procedure

The random search will be conducted with 64 trials in terms of each model. The hyperparameters need to be tuned according to their probability distributions are presented in Table I. Other hyperparameters not in the table are set to default values.

FFNNs and GBM were trained for 50000 epochs. For FFNNs and GBM, reducing the learning rate as the training progresses is useful to improve the learning ability. To this end, we use the schedule of reducing the learning rate when a metric stopped improving, commonly known as ReduceLRonPlateau. In this paper, the metric is chosen as the Mean Absolute Error (MAE) on the validation set. The learning rate is starting from 0.1 with patience of 100 epochs and decay factor of 0.5. The learning rate can be adaptively adjusted by this algorithm and need not to be manually tuned.

TABLE I: Hyperparameters optimized in random search

Method	Hyperparameter	Range or grid	Distribution
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
	$p^{\dagger}$	{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

† Only used when weight function is 'inv_distance'.

B. Performance evaluation

Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are used to assess the prediction performance on d_{CPA} and t_{CPA}:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|t_i - y_i| \tag{13}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2} \tag{14}$$

where y_i is the forecast value and t_i is the actual value.

To evaluate the classification accuracy of CD, confusion matrix is introduced [20]. The classification result contains four possible cases for the predicted class and the actual class: True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN). TP and TN are correct decisions made. FP and FN are also known as type I error and type II error, respectively. The illustration is shown in Figure 4. Furthermore, we will introduce four measures including True Positive Rate (TPR), True Negative Rate (TNR), False Negative Rate (FNR) and False Positive Rate (FPR), which are respectively defined as follows:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{15}$$

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} \tag{16}$$

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN}+\mathrm{TP}} \tag{17}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} \tag{18}$$

C. CPA prediction results

Table II depicts the prediction performance of horizontal distance, vertical distance and time of CPA in terms of different ML algorithms. Compared with our previous study in 2D [18], the prediction performance seems much more better due to trajectory segmentation. The best value of each performance metric is bolded.

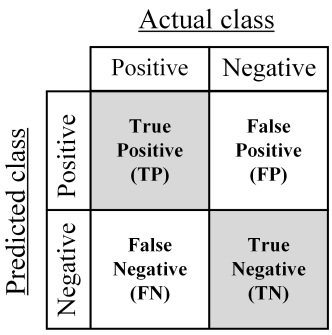


Figure 4: Confusion matrix visualization

TABLE II: Prediction results of $d_{\mathrm{CPA}_{xy}}$, d_{CPA_z} and t_{CPA}

Models	$d_{\mathrm{CPA}_{xy}}$(Nm)		d_{CPA_z}(ft)		t_{CPA}(s)	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
Baseline	4.52	9.63	29.01	203.3	42.73	67.83
FFNNs	0.30	0.56	9.32	66.85	**3.83**	**13.05**
KNN	1.56	2.13	12.88	95.96	13.67	30.85
GBM	**0.20**	**0.43**	**7.35**	**49.22**	4.63	13.39
RF	0.51	0.80	20.13	90.22	5.08	15.24

Since the assumption on constant speed and heading is not guaranteed, that the prediction results of CPA baseline model on $d_{\mathrm{CPA}_{xy}}$ and t_{CPA} is not satisfying. Nevertheless, the prediction results on d_{CPA_z} is relatively accurate, considering that the vertical separation minimum is 1000ft. It is derived from the low fluctuation of vertical speed. The results indicate that the CPA baseline model is unreliable for CPA prediction in the actual operations. In view of ML models, it can be seen that all models perform much more better than CPA baseline model. Thereinto, GBM outperforms other models in terms of $d_{\mathrm{CPA}_{xy}}$ and d_{CPA_z}. Comparing with the CPA baseline model, the MAE of is reduced by 4.32Nm (95.58%) and the RMSE is reduced by 9.20Nm (95.53%) for $d_{\mathrm{CPA}_{xy}}$ prediction, the MAE of is reduced by 21.66ft (74.66%) and the RMSE is reduced by 154.08ft (75.79%) for d_{CPA_z} prediction. In view of t_{CPA} prediction, FFNNs perform best in terms of t_{CPA}. Comparing with the CPA baseline model, the MAE of is reduced by 38.90 s (91.04%) and the RMSE is reduced by 54.78 s (80.76%).

Actually, the prediction performances of FFNNs, GBM, RF are very close. KNN is inferior to other approaches, which could be attributed to its insufficient learning capabilities.

D. Conflict detection results

To evaluate the effectiveness of our models for CD in 3D airspace, we conduct CD by comparing 1 with the calculated cylindrical norm of relative position at CPA based on predicted $d_{\mathrm{CPA}_{xy}}$ and d_{CPA_z}. It is noteworthy here that the minimum vertical separation is alternative, which is 1000ft below FL290 and 2000ft above FL290. To handle this problem, since the

fluctuation of vertical speed is negligible, we estimate the altitude of CPA of 2 aircraft using kinematic method based on the relative vertical speed and the t_{CPA} predicted with GBM:

$$Z_{\mathrm{CPA}_i} = Z_{5_i} + v_{i,5} \cdot t_{\mathrm{CPA}} \tag{19}$$

$$Z_{\mathrm{CPA}_o} = Z_{5_o} + v_{o,5} \cdot t_{\mathrm{CPA}} \tag{20}$$

At CPA, if one aircraft is below FL290, another is above FL290, the minimum separation standard H is still set as 1000ft. Otherwise, if Z_{CPA_i} and Z_{CPA_o} are below FL290, $H = 1000$ft. If Z_{CPA_i} and Z_{CPA_o} are above FL290, $H = 2000$ft.

The classification results are summarized in Table III. Though negative cases are mostly correctly classified, it is obvious that the baseline model is not suitable to detect conflict in real operations. 27.38% TPR indicates that an unignored proportion of conflict cases are not successfully identified. It should be noted that FN is arguably more serious than FP, because FN incorrectly identifies conflict case as conflict-free, which may let ATCO underestimate the danger and ignore it rather than deliver conflict resolution instructions. In view of ML models, GBM performs best in terms of TP and FN. Most conflict cases are correctly identified. Nevertheless, The FPR is a little bit high. That is to say, a little proportion of safe cases are classified as conflicts. FFNNs have the lowest FPR among all models, but the TPR is lower than GBM. FFNNs and GBM are alternative to be chosen, depending on the need for higher TPR or TNR. RF performs ordinarily among ML models. The CD performance of KNN is close to baseline model, which is also undesirable. Such results are capable of indicating that suitable ML models are able to strongly enhance the performance of CD.

TABLE III: Confusion matrix

Models	TP		FN		TN		FP	
	Num	Rate	Num	Rate	Num	Rate	Num	Rate
Baseline	18543	72.62%	6992	27.38%	221191	99.35%	1453	0.65%
FFNNs	24088	94.33%	1447	5.67%	222148	99.78%	496	0.22%
KNN	18742	73.40%	6793	26.60%	217772	97.81%	4872	2.19%
GBM	24914	97.57%	621	2.43%	214018	96.13%	8626	3.87%
RF	23917	93.37%	1618	6.34%	212627	95.50%	10017	4.50%

6. Conclusion

Air traffic growth remains strong over the next decades. Dense traffic will bring intensified workload to human operators. High accuracy of trajectory prediction will definitely support ATCOs or pilots to act more efficiently to solve conflicts, it also plays an important role in future ATM system.

In this paper, we introduced a novel automated data-driven CD framework based on Machine Learning (ML) approaches to predict CPA in 3D airspace. We firstly introduced the 3D CPA model with the cylindrical norm. Then, we created a close-to-reality dataset from Mode-S observations. Feature set is generated based on the nature of this problem. Furthermore, prevailing ML methods were built to predict the time, horizontal distance and vertical distance of CPA. The prediction results were compared with the conventional baseline model.

The results indicate that: firstly, compared with our previous study in 2D [18], the prediction performance seems much better due to trajectory segmentation. Secondly, the CPA baseline model shows less accuracy for CPA prediction in actual operational environment. While, all used ML models perform much better than CPA baseline model, and GBM outperforms other ML models in terms of $d_{\mathrm{CPA}_{xy}}$ and d_{CPA_z}. Thirdly, in the CD results, 27.38% TPR with the baseline model indicates that an unignored proportion of conflict cases are not successfully identified. While with GBM and FFNNs models, the TPR is much lower, less than 5%. Thus, suitable ML models are able to strongly enhance the performance of CD.

References

[1] ICAO, *Doc 9574, Manual on Implementation of a 300 m (1000 ft) Vertical Separation Minimum Between FL 290 and FL 410 Inclusive*, Second edition, 2001.

[2] M. Prandini, J. Hu, J. Lygeros, and S. Sastry, "A probabilistic approach to aircraft conflict detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 199–220, 2000.

[3] D. G. Denery and H. Erzberger, *The Center-Tracon Automation System: Simulation and Field Testing*, 1997.

[4] D. J. Brudnicki, K. S. Lindsay, and A. L. Mcfarland, "Assessment of field trials, algorithmic performance, and benefits of the user request evaluation tool (uret) conflict probe," in *IEEE Digital Avionics Systems Conference*, 1997.

[5] P. Brooker, "Stca, tcas, airproxes and collision risk," *The Journal of Navigation*, vol. 58, no. 3, pp. 389–404, 2005.

[6] M. D. Ciletti and A. W. Merz, "Collision avoidance maneuvers for ships," *Navigation*, vol. 23, no. 2, pp. 128–135, 1976.

[7] E. M. Goodwin, "A statistical study of ship domains," *Journal of Navigation*, vol. 28, no. 3, pp. 328–344, 1975.

[8] P. Bauer, A. Hiba, J. Bokor, and A. Zarandy, "Three dimensional intruder closest point of approach estimation based-on monocular image parameters in aircraft sense and avoid," *Journal of Intelligent & Robotic Systems*, pp. 1–16, 2018.

[9] J. Kucher and L. Yang, "Survey of conflict detection and resolution modeling methods," in *AIAA Guidance, Navigation, and Control Conference*, 1997, pp. 1388–1397.

[10] C. Munoz, A. Narkawicz, and J. Chamberlain, "A tcas-ii resolution advisory detection algorithm," in *Aiaa Guidance, Navigation, & Control*, 2013.

[11] L. C. Yang and J. K. Kuchar, "Prototype conflict alerting system for free flight," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 4, pp. 768–773, 1997.

[12] Y. Huo, D. Delahaye, and Y. Wang, "Sensitivity analysis of closest point of approach," in *ICRAT 2018, 8th International Conference for Research in Air Transportation*, 2018.

[13] S. Alam, H. Abbass, C. Lokan, M. Ellejmi, and S. Kirby, "Computational red teaming to investigate failure patterns in medium term conflict detection," in *8th Eurocontrol Innovation Research Workshop, Eurocontrol Experimental Center, Brtigny-sur-Orge, France*, 2009.

[14] Q. Yang, A. Lim, K. Casey, and R.-K. Neelisetti, "An enhanced cpa algorithm for real-time target tracking in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 5, no. 5, pp. 619–643, 2009.

[15] C. A. Munoz and A. J. Narkawicz, "Time of closest approach in three-dimensional airspace," 2010.

[16] G. Dowek and C. Munoz, "Conflict detection and resolution for 1, 2,... n aircraft," in *7th AIAA ATIO Conf, 2nd CEIAT Int'l Conf on Innov and Integr in Aero Sciences, 17th LTA Systems Tech Conf; followed by 2nd TEOS Forum*, 2007, p. 7737.
[17] Z. Wang, M. Liang, and D. Delahaye, "A hybrid machine learning model for short-term estimated time of arrival prediction in terminal manoeuvring area," *Transportation Research Part C: Emerging Technologies*, vol. 95, pp. 280–294, 2018.
[18] Z. Wang, M. Liang, D. Delahaye, and W. Wu, "Learning Real Trajectory Data to Enhance Conflict Detection Accuracy in Closest Point of Approach," in *ATM Seminar 2019, 13th USA/Europe ATM R&D Seminar*, Vienne, Austria, Jun. 2019.
[19] S. Arumugam and C. Jermaine, "Closest-point-of-approach join for moving object histories," in *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*. Citeseer, 2006, pp. 86–86.
[20] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.
[21] B. C. Csáji, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, p. 48, 2001.
[22] C. M. Bishop, "Pattern recognition and machine learning (information science and statistics)," no. 4, p. 049901, 2006.
[23] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.
[24] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.
[25] 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.
[26] 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.
[27] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[28] 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.

AUTHORS BIOGRAPHY

Zhengyi Wang received the B.E. degree in Flight Vehicle Propulsion Engineering from Civil Aviation University of China (CAUC) in 2016, the M.Sc. degree in Operational Research from École Nationale de l'Aviation Civile (ENAC) in 2018, the M.Eng. in Aviation Engineering and "titre d'ingénieur diplômé" from CAUC in 2019. He is currently a Ph.D. candidate at ENAC.

Man (Annie) Liang is currently a lecturer in School of Engineering, University of South Australia (UniSA). She obtained her Ph.D in Applied Mathematics from EDAA, University of Toulouse&ISAE-SUPAERO in Feb. 2018 and did a post-doc at ENAC. During 2006-2014, she worked at the department of Air Traffic Management at CAUC as a lecturer and researcher. She holds the Chinese air traffic controller license.

Daniel Delahaye is working as a Professor and Director in the OPTIM Laboratory of ENAC. He obtained his engineering degree from the ENAC school and a master of science in signal processing from the National Polytechnic Institute of Toulouse in 1991. He obtained his Ph.D in automatic control from the Aeronautic and Space National school in 1995 and did a post-doc at the Department of Aeronautics and Astronautics at MIT in 1996.

APPENDIX

1) Feed-Forward Neural Networks: A specific class of NNs is introduced to approximate the function $f(\cdot)$, referred to as Feed-Forward Neural Networks (FFNNs). Feed-Forward Neural Networks (FFNNs) are the first and simplest type of NNs. It is said to be universal functional approximators [21]. The feed-forward term means that the architecture doesn't have closed directed cycles, which ensures that the outputs are deterministic functions of the inputs [22]. Given input vector $(\mathbf{x}_1, ..., \mathbf{x}_D)^T$, the output can be calculated as:

$$y(\mathbf{x}; \mathbf{w}) = \Psi\left(\sum_{i=1}^{M} w_i \Phi\left(\sum_{j=1}^{D} w_{ji}\mathbf{x}_j + w_{j0}\right) + w_0\right) \quad (21)$$

where M is the hidden layer node number, w_{ji} is the weight between the j-th input node and the i-th hidden node, w_i is the weight between the i-th hidden node and the output node, w_{j0} is the bias to the i-th hidden layer, w_0 is the bias to the output layer. Frequently used Φ including Rectified Linear Unit (ReLU) function, hyperbolic tangent (tanh) function and sigmoid function. they are respectively defined as follows:

$$\Phi_{\mathrm{ReLU}}(x) = \max(0, x) \quad (22)$$

$$\Phi_{\tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (23)$$

$$\Phi_{\mathrm{sigmoid}}(x) = \frac{1}{1 + e^{-x}} \quad (24)$$

Ψ is the identity function:

$$\Psi(z) = z \quad (25)$$

2) K-Nearest Neighbors: K-Nearest Neighbors (KNN) [23] is a non-parametric and a lazy learning algorithm. In the regression analysis, KNN predict the unseen instance by local interpolation of the K nearest neighbors in the training set. The pseudo-code of KNN is given in Algorithm 3.

Algorithm 3 KNN for regression

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

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 $\mathbf{x}$ based on $\{d(\mathbf{X}_n, \mathbf{x})\}_{1\le n\le N}$
  $y \leftarrow \sum_{n\in N_K} \mathbf{Y}_n w(d(\mathbf{X}_n, \mathbf{x}))$
  return y
end procedure

where $d(\cdot, \cdot)$ is the distance function and $w(\cdot)$ is the weight function.

The distance metric of KNN provides a way to describe similarity between examples. The prediction can be greatly improved by choosing an appropriate distance metric. Minkowski distance is widely used:

$$d(\mathbf{m}, \mathbf{n}) = \left(\sum_{j=1}^{D} \left| m_j - n_j \right|^p \right)^{1/p} \tag{26}$$

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.

The weight function has two frequently used types: uniform weighting and inverse distance weighting.

Uniform weighting function assuming all neighbors are weighted equally:

$$w(d(\mathbf{X}_n, \mathbf{x})) = \frac{1}{card(N_K)}, \quad \forall n \in N_K \tag{27}$$

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

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

$$w(d(\mathbf{X}_n, \mathbf{x})) = \frac{1/d(\mathbf{X}_n, \mathbf{x})}{\sum_{m \in N_K} 1/d(\mathbf{X}_m, \mathbf{x})}, \quad \forall n \in N_K \tag{28}$$

3) Gradient Boosting Machine: Gradient Boosting Machine (GBM) is a famous ensemble learning method and can be viewed as iterative functional gradient descent algorithms [24].

Given a training set $\{(\mathbf{x}_1, t_1), ..., (\mathbf{x}_N, t_N)\}$ and expected number of weak learners M, initializing the model as:

$$F_0(\mathbf{x}) = \arg\min_{\gamma} \sum_{n=1}^{N} L(t_n, \gamma) \tag{29}$$

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

$$r_{nm} = -\left[\frac{\partial L(t_n, F(\mathbf{x}_n))}{\partial F(\mathbf{x}_n)} \right]_{F(\mathbf{x}) = F_{m-1}(\mathbf{x})}, \quad n = 1, ..., N \tag{30}$$

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

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \gamma_m h_m(\mathbf{x}) \tag{31}$$

where the multiplier γ_m is computed by:

$$\gamma_m = \arg\min_{\gamma} \sum_{n=1}^{N} L\left(y_n, F_{m-1}(\mathbf{x}_n) + \gamma h_m(\mathbf{x}_n)\right) \tag{32}$$

Gradient Boosting Decision Tree (GBDT) is a frequently used boosting type of GBM. Other methods include Dropouts meet Multiple Additive Regression Trees (DART) [25], Gradient-based One-Side Sampling (GOSS) [26], etc.

4) Random Forests: Random Forests (RF) [27] 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. With the combination of weak learners, a stronger learner will be generated. As an ideal candidate 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 too much. In addition, because the law of large numbers, overfitting is seldom seen in RF with sufficient number of data. The pseudo-code of RF is given in Algorithm 4 [28].

Algorithm 4 RF for regression

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

Output:
$y \in \mathbb{R}$: Output of $\mathbf{x}$

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