

Article

Feasibility of Conflict Prediction of Drone Trajectories by Means of Machine Learning Techniques

Victor Gordo ^{1,2,*} , Javier A. Perez-Castan ¹ , Luis Perez Sanz ¹ , Lidia Serrano-Mira ¹  and Yan Xu ³

¹ Sistemas Aeroespaciales, Transporte Aéreo y Aeropuertos, Universidad Politécnica de Madrid (UPM)—ETSI Aeronáutica y del Espacio, 28040 Madrid, Spain; javier.perez.castan@upm.es (J.A.P.-C.); l.perez@upm.es (L.P.S.); lidia.serrano@upm.es (L.S.-M.)

² Ingeniería y Economía del Transporte (INECO), 28036 Madrid, Spain

³ School of Electronic and Information Engineering, Beihang University, Beijing 100191, China; yan.xu@buaa.edu.cn

* Correspondence: vgordo@ineco.com or victor.gordo@upm.es

Abstract: The expected number of drone operations in the coming decades, together with the fact that most of them will take place in very-low-level airspace, will lead to a density of drone flights much greater than that of conventional manned aviation. In this context, the number of conflicts (i.e., 4D convergence of drone trajectories below the safe separation minima) will be much more frequent than in manned aviation and, therefore, conventional air traffic management methods or even the specific proposed mechanisms for drone traffic management are unlikely to be able to solve them safely. This paper considers a set of simulated drone trajectories in a high-density urban environment to analyze the applicability of machine learning regression and classification techniques to detect conflicts among such trajectory times in advance of their occurrence in order to provide new methods to manage the expected drone traffic density safely and efficiently. This would not be possible with current drone traffic management solutions. The obtained results suggest that the Random Forest, Artificial Neural Networks and Logistic Regression algorithms could detect nearly all near-collisions up to 10 s before they occur, and the first two algorithms could also detect a significant number of near-collisions more than 60 s earlier.

Keywords: UAS; drone; conflict; machine learning; classification; congested; airspace; U-space; tactical



Citation: Gordo, V.; Perez-Castan, J.A.; Perez Sanz, L.; Serrano-Mira, L.; Xu, Y. Feasibility of Conflict Prediction of Drone Trajectories by Means of Machine Learning Techniques. *Aerospace* **2024**, *11*, 1044. <https://doi.org/10.3390/aerospace11121044>

Academic Editors: Sebastian Schier-Morgenthal and Michael Schultz

Received: 25 September 2024

Revised: 3 December 2024

Accepted: 4 December 2024

Published: 20 December 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The proliferation of Unmanned Aircraft Systems (UASs), popularly known as drones, in the civil market has created a myriad of applications [1], as they can be applied to numerous professional activities, such as surveying and mapping, precision agriculture, infrastructure monitoring, public safety, delivery of goods, etc. [2,3]. As such, market forecast studies predict a great number of UAS operations in the coming decades [3–5].

Among all the use cases, last-mile drone delivery is expected to be one of the most demanded ones. Its market potential in European cities was studied in [6], calculating the costs and returns associated with delivering a parcel with a drone based on the number of people within the delivery range of potential drone hives across the EU, considering different technological scenarios. The results suggested that at least 7% of the EU citizens could benefit from a viable last-mile drone-delivery service and up to 30% from technological improvements on existing drones. The viability of this service would be concentrated in densely populated urban areas, such as in the UK, Germany, Italy, and France, i.e., countries where drone hives may undergo the most efficient development. Translating this demand into the number of drone operations, this study calculated 21 to 109 drone trips per hour in the Paris drone hive alone for small parcel delivery from a single hive.

These figures are credible, since several companies are already developing drone delivery, and large growth is expected in the near future. Wing, a subsidiary company of

Alphabet Inc., Mountain View, CA, USA, has already deployed six drone delivery facilities in Dallas Fort-Worth (TX, USA), and, in partnership with Walmart, they have performed more than 350,000 deliveries to date [7]. In Europe, Manna Aero has already performed more than 100,000 drone deliveries in Ireland, and they foresee the development of over 1000 deliveries per day in individual dense urban markets, which also seems to be credible, as they expect to be performing up to 2000 to 3000 flights per day across their multiple locations [8].

Similarly, Amazon Prime Air has already received approval from the Federal Aviation Administration (FAA) to operate their drones beyond visual line of sight (BVLOS) in College Station (TX, USA), thanks to the on-board detect-and-avoid technology. The company has the goal of delivering 500 million packages by drone, worldwide, before 2030 [9].

Therefore, once the existing technological and regulatory barriers have been solved [10], a large amount of small UAS operations can be expected in dense urban areas, especially considering that the cost associated with UAS operations tends to be very small compared with that of the current method; for example, the cost of a drone delivery can be estimated as one-tenth that of the cost of a human driver in a car [8].

Expanding the scope to urban air mobility (UAM), market studies foresee that, together with a commercial market for last-mile delivery operations already emerging by 2030, up to 25,000 eVTOL will be operating worldwide by 2035, mainly in metropolitan areas, with demand raising exponentially from 2025 to 2040 [11–14].

Experience with electric vehicles shows, however, that the forecasted adoption rates from researchers were higher than the real ones [15], which suggests the need to be cautious with UAS and UAM forecasts, especially if the introduction of this technology does not include an in-depth analysis about how potential consumers could feel more comfortable with it. However, we cannot disregard the fact that, in urban areas, the global demand for passenger mobility will double by 2050, and the growth will be even greater for goods mobility [16]. In this context, UAM will be essential to alleviate overwhelmed ground transportation systems.

Therefore, we can expect that the number of UASs and, in general, UAM operations will be much greater than in manned aviation. Considering that most of these operations will take place in very-low-level (VLL) airspace, i.e., below 1000 ft above the highest obstacle over cities and congested areas and 500 ft elsewhere, to segregate them from manned aircraft as per International Civil Aviation Organisation (ICAO) Rules of the Air [17], it can be assumed that the density of operations in the airspace will be several orders of magnitude higher than in manned operations [2].

Introducing airspace structures could minimize complexity but would limit the degrees of freedom and reduce the overall capacity. Therefore, non-structured airspaces will be desirable for maximizing capacity and allowing the most efficient trajectories for air vehicles while, at the same time, assuming the cost of a greater probability of conflicts among operations, i.e., distances below predefined safe separation minima, which will require intervention to avoid potential collisions [18].

Accordingly, to manage this high density of UAS operations, from the early stages of civil UAS operations, the US and the EU envisioned that UAS traffic management (UTM) systems would be essential for keeping safety risk below acceptable thresholds. The FAA developed a concept of operations for UTM [2], and in Europe, the U-space concept was formulated [19]. The main objective of UAS traffic management services is to limit, to an acceptable level, the risk of collision of UASs, i.e., they are focused on managing UAS conflicts. In ATM [20], conflict management can be based on three layers: (1) strategic conflict management, (2) separation provision and (3) collision avoidance.

Strategic Conflict Management prevents conflicts before the execution of the flights by means of airspace structures, dynamic capacity balance, and flight plan synchronization; however, as explained, airspace structures limit capacity, so they are generally disregarded in UAS traffic management.

Tactical Conflict Management (separation provision) is based on ensuring that UASs keep a safe separation distance during the flight execution, i.e., they remain well clear from one another. To achieve this, tactical conflict management relies on the real-time positions provided by surveillance and tracking services to inform operators of potential deviations from their approved 4D flight plan and informing them about the proximity of other UASs. Finally, collision avoidance is activated when the safe separation minima among UASs is broken; this is generally outside of the scope of the UAS traffic management services.

However, in the EU, a first regulation for U-space airspaces was mandated in 2021 [21]; however, it only considers the strategic deconfliction layer, not accepting any new UAS flight authorization requests if the flight under that UAS flight authorization is not free of intersection in space and time with any other notified UAS flight authorizations within the same U-space airspace.

In the same way, the UTM original concept of operations is based on a strategic deconfliction [22] concept in which operations should be free of 4D intersection with all other known UAS operations prior to departure. In fact, the current standard for UTM implementation developed by the ASTM (American Society for Testing and Materials) [23] only prevents UAS conflicts by means of strategic conflict detection, considering tactical deconfliction to be outside of its scope for the time being. This implies that this standard imposes that operational intents, i.e., UAS flight plans, cannot intersect in space and time, which implies that the spatial dimensions of the 4D volumes of two different flight plans cannot share even one point if the start/end time range for the two 4D volumes overlap.

The greater the volumes demanded by flight plans and, especially, the greater the uncertainty regarding the time of execution, the larger the occupied 4D volumes will be and, therefore, the lower the available capacity and number of simultaneous UAS operations. This impact has already been shown in UTM experiments, which evidence that a large volume size, especially when there are crossing points, leads to a reduction in the density of flights passing through those crossing points [24].

Therefore, this approach is considered only for the initial deployment of U-space/UTM, but it is clear that it will not be able to manage the expected number of UAS operations. Accordingly, new strategies for strategic deconfliction were soon proposed, for example the lane-based method [25], to reduce delays and improve capacity. More advanced approaches are currently being explored to enlarge capacity.

However, although advanced strategic deconfliction techniques could help to handle a larger number of UAS operations, thereby increasing capacity, they will have an important impact on efficiency, increasing delays on the ground and reducing the throughput of demanded UAS operations [26]. For example, in an experiment with a demand of 500 operations per hour, it was shown that, even in the most favorable case, the real throughput (operations/hour) would not reach more than 20% of the requested operations, leading to unrealistically high delays on the ground.

Similarly, the authors of [27] showed that, in a congested airspace with 300 airspace operations per hour or more, autonomous algorithms for flight plan strategic deconfliction can be used to reduce tactical conflicts to almost zero, improving safety, albeit at the cost of significantly affecting the mission completion time.

In this context, if all the demanded operations need to be developed when required, the number of tactical conflicts (i.e., 4D convergence of UAS trajectories below a safe separation minima) will be much more frequent than in manned aviation and, therefore, conventional air traffic management methods are unlikely to be able to solve them safely. Meanwhile, the U-space Tactical Conflict Detection services developed so far are very immature and unable to handle such a large number of potential tactical conflicts and detect them with sufficient time to be solved in every case.

To fill this gap, this paper analyzes the applicability of machine learning classification techniques in the tactical phase, i.e., during flight, without previous strategic deconfliction of UAS trajectories, to predict these conflicts times before their occurrence, providing new methods to safely and efficiently manage large UAS traffic densities. Previous research

has already explored the applicability of machine learning (ML) techniques to predict UAS flight times and short-term trajectory predictions for a specific UAS model based on the performance of the UAS and weather conditions [28,29]. That research demonstrated the accuracy of a method developed to predict the flight-time required by a drone to complete a planned path, adopting a machine-learning-based approach. The second paper even adapted the method for conflict detection of two crossing paths. However, the method requires a detailed flight plan of the proposed operations and an extensive dataset of the specific UAS model performing the operation.

The novelty of the present paper consists of exploring the ability of ML to predict conflicts among a large number of different UAS flying simultaneously, based only on the positions reported during their flights and without knowing previously the planned trajectory of each drone, i.e., without previous flight planning of the operations in the UAS traffic management system, followed by a strategic deconfliction process, because, as previously explained, this would lead to big delays. This approach would make it possible to develop the most efficient trajectories for UAS operators, without delays. However, the challenge is to detect all the potential conflicts in advance in order to ensure safety. The reason for the application of ML instead of analytical techniques is that the later makes it possible to estimate conflicts if the bearing of the aircraft remains constant [30], but UASs, in particular multirotors, can change their trajectories very fast and, therefore, it is necessary to detect conflicts even when the bearing changes suddenly. ML techniques, especially ensemble models, can detect complex patterns, so they have the potential to detect conflicts before two UASs start following a bearing that will lead them to their closest point of approach.

In summary, this paper analyzes the feasibility of detecting UAS conflicts in flight for a large number of concurrent operations by means of machine learning classification techniques to overcome the limitations of existing UAS traffic management methods. Section 2 describes the methodology used to predict conflicts between UAS pairs and the dataset considered to test it. Section 3 presents the results of the applied ML algorithms and, finally, Section 4 shows the main conclusions.

2. Materials and Methods

As explained above, it is expected that, in the near future, the density of UAS operations will be much higher than that of manned aircraft operations [2], necessitating new methods to handle this number of operations safely and efficiently. The baseline scenario considered is the reference scenario modelled in DACUS D4.2 Experiment 4 [31], which represents the expected drone traffic in the city of Madrid on a typical day by the year 2035. This scenario foresees a wide range of UAS operations, from very small drones for surveillance tasks (less than 1 m in length and less than 2 kg in weight) to delivery drones of varying sizes (with maximum take off masses (MTOMs) of up to 18 kg), as well as large passenger-carrying air taxis (up to 13 m in length and 5600 kg MTOM). Figure 1 shows the number of UAS operations per timeframe in the baseline scenario.

The maximum number of UAS operations is expected to occur in the 14:00 to 15:00 timeframe. Hence, this timeframe will be considered hereafter to evaluate the applicability of machine learning techniques to predict conflicts between UAS trajectories. In this timeframe, 742 UAS operations are expected with a total flight time of 410 h. The trajectories of these operations are shown in Figure 2.

These operations would take place between several vertiports in the city of Madrid and logistics centers in the suburbs, as shown in Figure 3. This figure also indicates the number of operations per vertiport, with each vertiport location being represented by a dot, colored according to the number of drone operations carried out there.

To evaluate the density of operations, the area overflowed by the UAS has been divided into square cells of 4.14 km². Figure 4a,b present density maps of the total time and distance flown during this period considering the defined grid. These figures show that, in a single

hour, more than 20 flight hours and more than 450 km of drone flights would take place in the busiest cells of the city center.

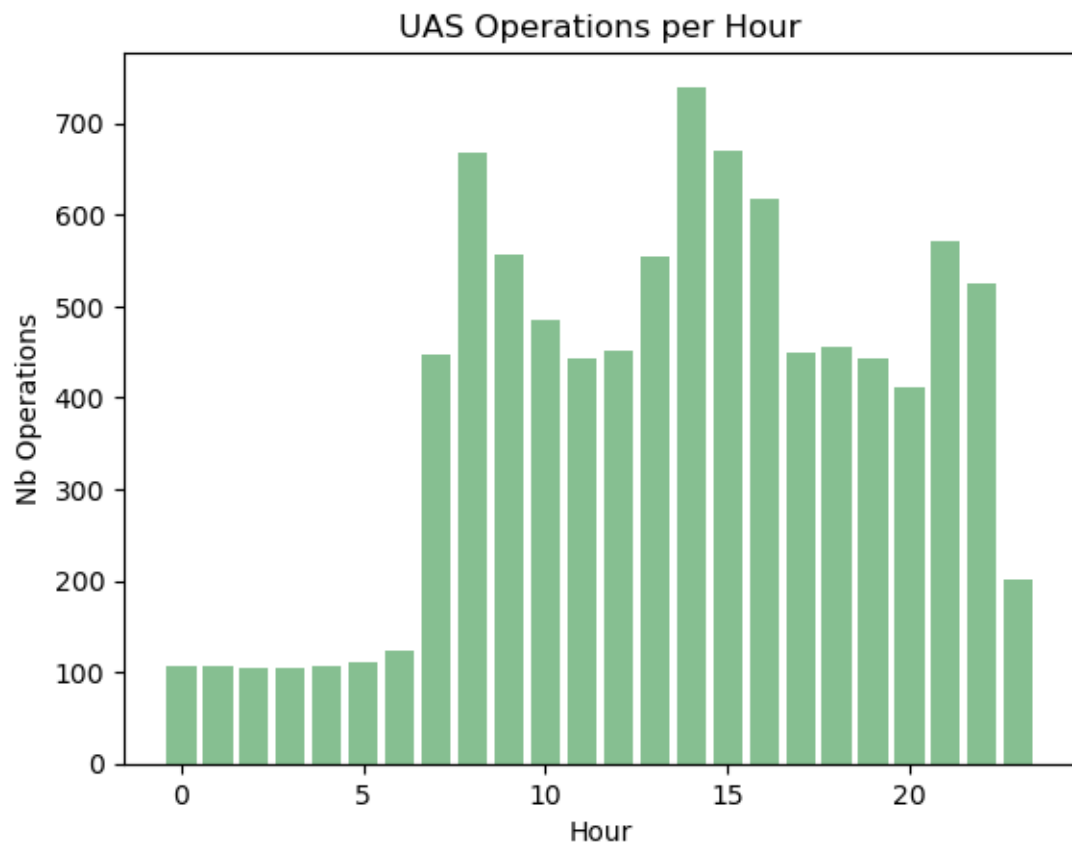


Figure 1. Drone traffic density in the reference scenario per timeframe on a typical day.

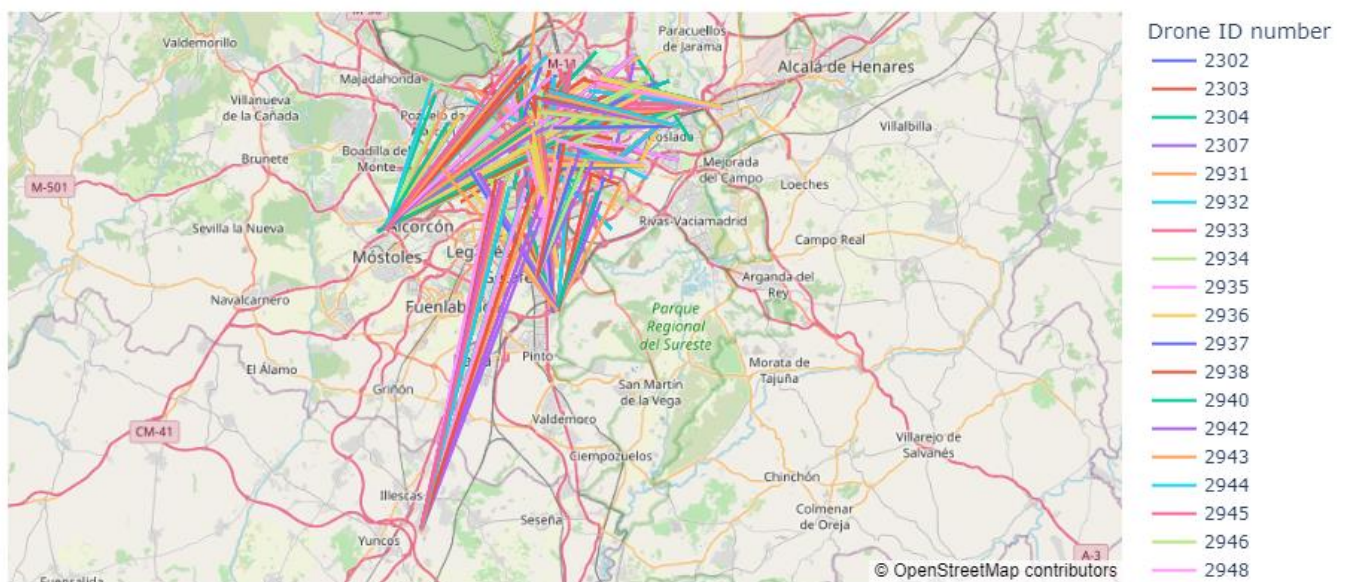


Figure 2. UAS trajectories in the 14H–15H timeframe.

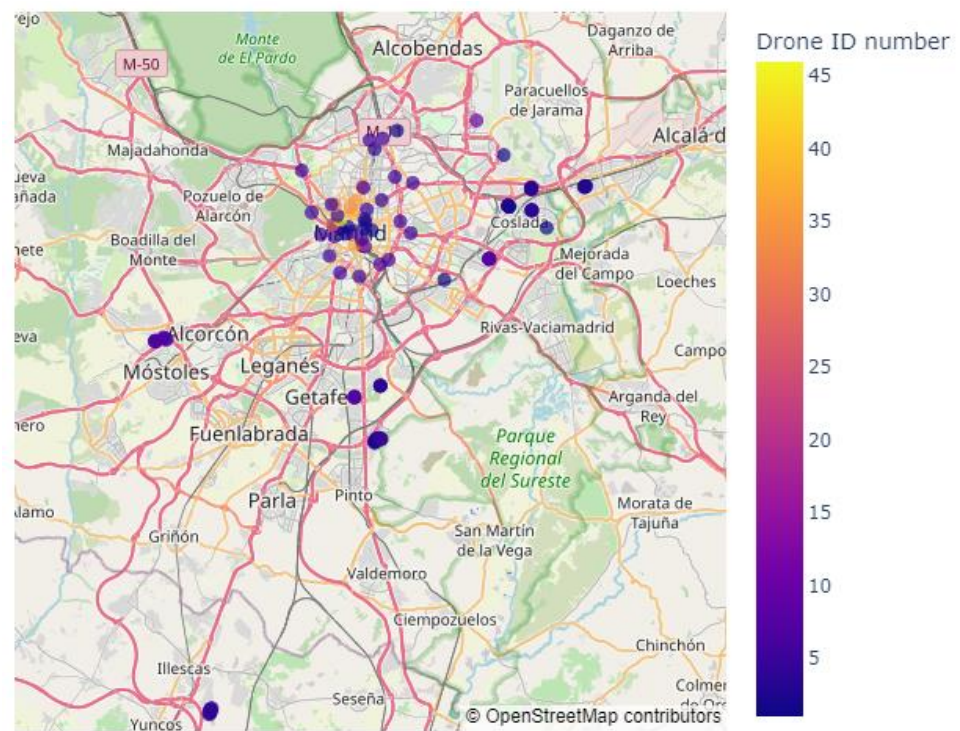


Figure 3. Operations per vertiport in the 14H–15H timeframe.

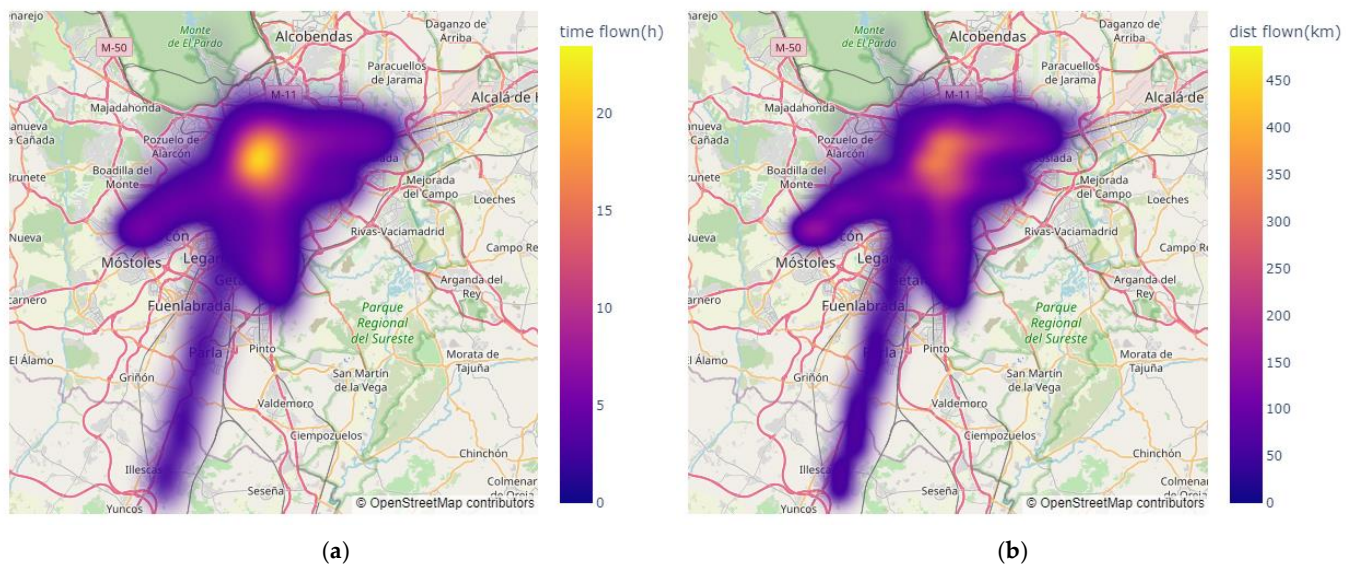


Figure 4. Total time flow (a) and total distance flow (b) per cell in the 14H–15H timeframe.

Managing this density of operations with the existing UTM/U-space implementations [21,22] would imply a large number of strategic conflicts identified by the Strategic Conflict Resolution (SCR) service. This service checks that the 4D trajectories, containing the flights for 95% of the time, of UAS flight authorization requests do not overlap with other already approved UAS flight authorizations. Therefore, the SCR would report a large number of operations that would not be approved or would have to be modified in time and/or trajectory.

For example, Figure 5 shows the 4D occupancy of the cells in the defined grid, taking into account the horizontal and vertical 95% buffers of UAS operations, with respect to the total flight distance, proposed by the SESAR BURDI project [32], i.e., 120 m and 25 m, respectively, and the minimum time buffer of 10 min, as defined by the ASTM

standard F3548-21 [23]. It can be seen that, in several cells, the total occupancy would be higher than 100%, i.e., in these cells, the SCR service would have to reject a part of the requested operational intents. Even in cells with a total occupancy below 100% but above 50%, it is likely that some of the operational intents would be rejected by the SCR due to 4D trajectories overlapping with other operational intents in these cells. As a result, a significant proportion of the requested operations would have to be rejected, modified or delayed, resulting in inefficiencies.

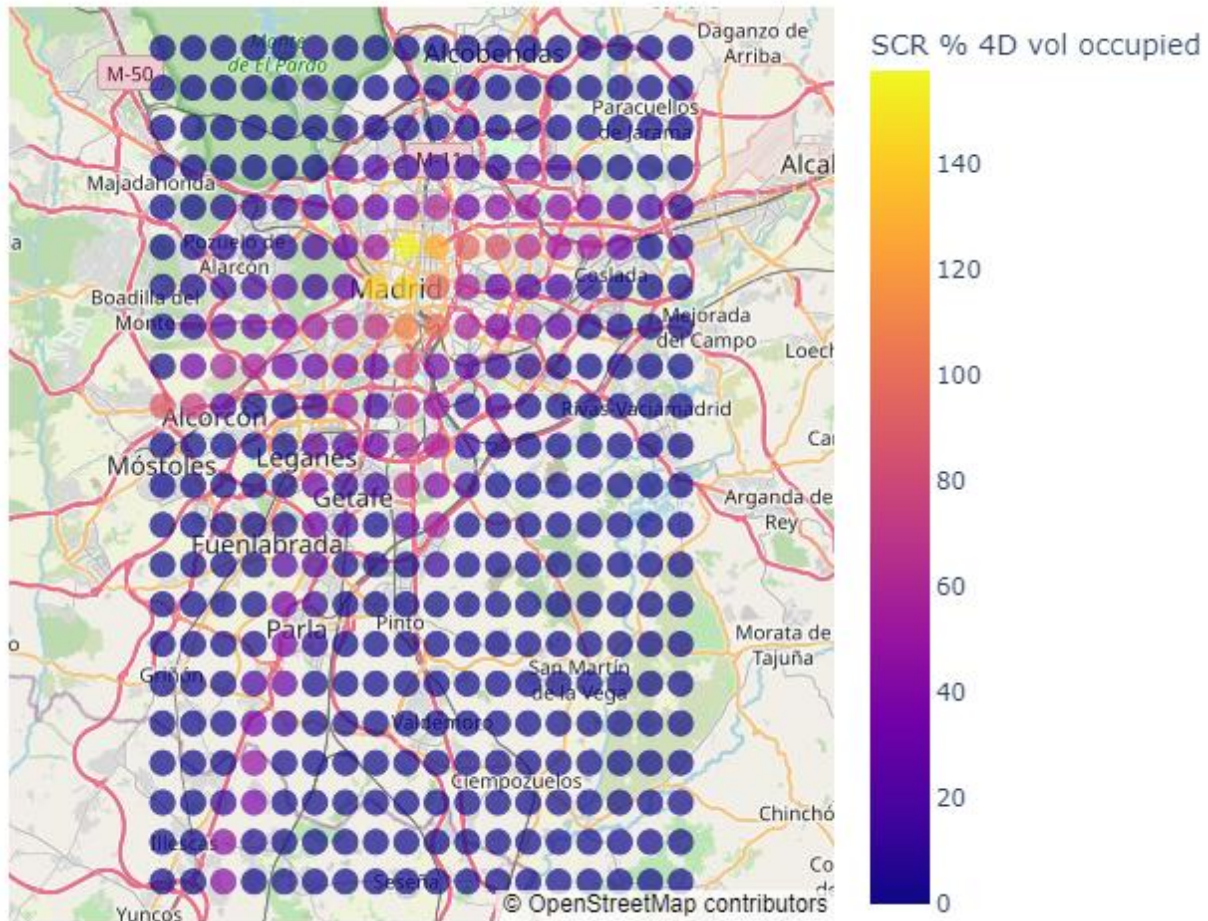


Figure 5. 4D occupancy per cell in the 14H–15H timeframe.

To overcome these problems, this article explores the use of machine learning techniques to identify potential conflicts during the flight without prior strategic deconfliction, in order to enable the development of all required operations. To feed these algorithms, the initial dataset was modified to present the minimum distance between UAS pairs. To obtain this modified dataset, the distance between each pair of UASs was calculated every second during the 14H to 15H timeframe, using the series of positions and timestamps of each UAS provided by the original dataset. Then, the distances per second of each UAS pair were analyzed to identify the instant of closest proximity, and this second of the day together with the minimum distance, were recorded in the modified dataset for each UAS pair.

To calculate the distances between pairs of UASs, we have considered the position and speed vector of an UAS at a given time:

- $x_i(t_0)$: Position of UAS “i” in the east–west direction
- $y_i(t_0)$: Position of UAS “i” in the north–south direction
- $z_i(t_0)$: Altitude of UAS “i”
- $v_i(t_0)$: Velocity of UAS “i”
- $\theta_i(t_0)$: Heading in the horizontal plane (x,y) of UAS “i”

- $\phi_i(t_0)$: Pitch in the vertical plane (y,z) of UAS “i”.

Accordingly, the distance D between any UAS pair at that instant of time, t_0 , is:

$$D_{ij}(t_0) = \sqrt{(x_i(t_0) - x_j(t_0))^2 + (y_i(t_0) - y_j(t_0))^2 + (z_i(t_0) - z_j(t_0))^2} \quad (1)$$

Next, we classified the minimum distance at the closest point of approach for each pair of UASs as a near-collision if it was lower than a predefined threshold, which, in this paper, was set at 50 m, although other thresholds could be applied. This near-collision distance is defined as a situation where a pair of UASs would require an avoidance maneuver to prevent an accident. The optimal distance threshold for detecting a near-collision will depend on a number of factors, like the speeds and sizes of the UAS, the navigation and flight technical errors, etc. This paper considered 50 m as a baseline, but future research will explore the impact of different thresholds on the ability of machine learning algorithms to detect conflicts between UASs.

As the original dataset was composed of 742 UAS, in the 14H to 15H period, the number of potential pairs would be 274,911 ($N \times (N - 1)/2$). However, the number of near-collisions was much lower, i.e., only 1335 of all the potential pairs; in other words, the dataset was extremely imbalanced. Therefore, we disregarded the UAS pairs whose minimum distance was greater than 1000 m, obtaining a final dataset of 13,869 pairs, of which only 1335 pairs were classified as near-collisions.

Finally, for each pair of UAS in the dataset, we calculated the distances 1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90, and 100 s before the closest point of approach; longer times are disregarded in this article.

In addition to the real distances (dreal) for each pair at these instants of time, we also calculated the minimum projected distances (dproj) at each of these instants, i.e., the minimum distance the UAS pair would reach if they continued flying from that point indefinitely with the same bearing and speed vector.

If both UASs continued lineally their trajectories, the distance at later instant t between UAS i and j would be:

$$D_{ij}^2(t) = D_{ij}^2(t_0) + V_{rel_{ij}}^2 t^2 + 2Bt \quad (2)$$

where

$$V_{rel_{ij}} = \sqrt{v_i^2 + v_j^2 - 2v_i v_j \cos\phi_i \cos\phi_j (\cos(\Delta\theta) + \sin\phi_i \sin\phi_j)} \quad (3)$$

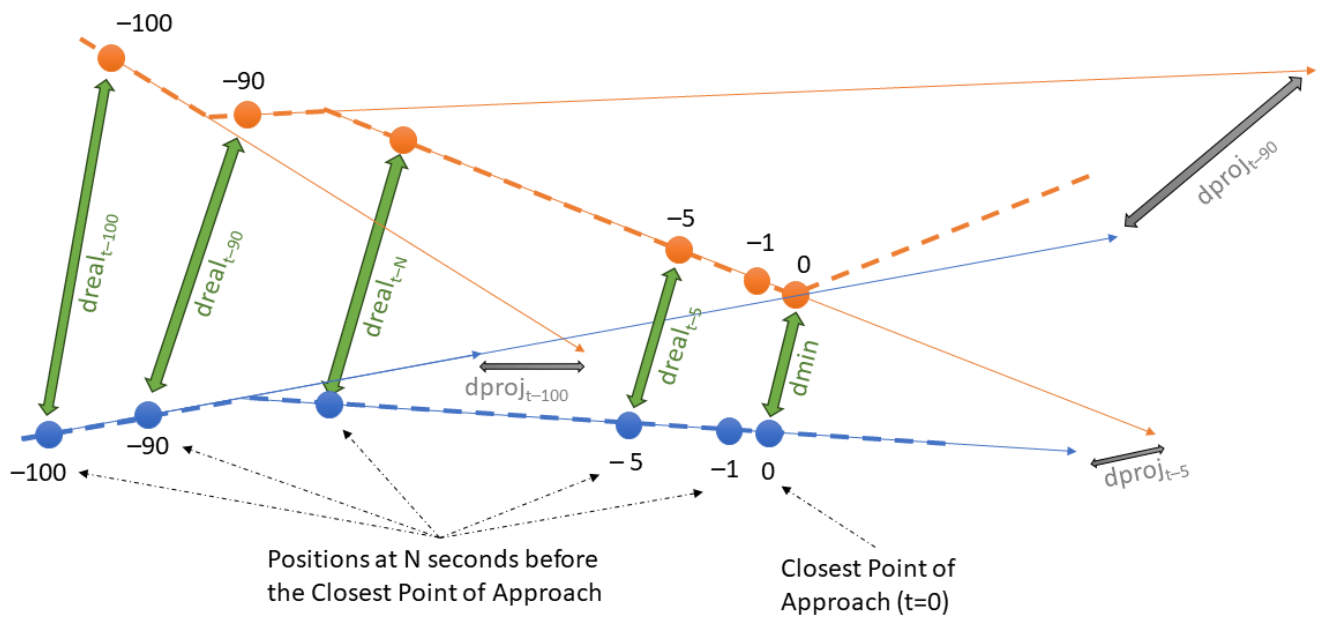
and

$$B = \Delta x(t_0)(v_i \cos\phi_i \cos\theta_i - v_j \cos\phi_j \cos\theta_j) + \Delta y(t_0)(v_i \cos\phi_i \sin\theta_i - v_j \cos\phi_j \sin\theta_j) + \Delta z(t_0)(v_i \sin\phi_i - v_j \sin\phi_j) \quad (4)$$

Then, the minimum projected distance for any pair of UASs continuously following their speed vectors from their initial positions can be calculated as follows:

$$\frac{\partial D_{ij}^2(t)}{\partial t} = 0 \rightarrow t_{min_{proj}} = \frac{-B}{V_{rel_{ij}}^2} \rightarrow D_{ij}(t_{min_{proj}}) = \sqrt{D_{ij}^2(t_0) - \frac{B^2}{V_{rel_{ij}}^2}} \quad (5)$$

The concept of real and projected distances is described in Figure 6, where two UAS trajectories are represented by dashed lines (blue and orange). It can be seen that real distances tended to decrease, especially when the UASs were reaching their closest point of approach, but projected distances varied if the bearing of the UASs changed.



$dreal_{t-N}$: real distance among the UAS pair, N seconds before the Closest Point of Approach

$dproj_{t-N}$: minimum distance that the UAS pair would reach if both UAS continue flying with the same heading and speed indefinitely, calculated for the situation at N seconds before the Closest Point of Approach

Figure 6. Illustration of real (dreal) and projected (dproj) distances along two UAS trajectories.

A sample of the final obtained dataset can be seen in Table 1.

Table 1. Sample of UAS pair minimum distances and time of occurrence and distances (real and projected) at previous instants of time.

UAS1	UAS2	t_{dmin} (Sec of Day) ¹	$dmin$ (m) ²	Near Collision	$dreal_{t_{dmin}-1}$ (m) ³	$dproj_{t_{dmin}-1}$ (m) ⁴	...	$dreal_{t_{dmin}-90}$ (m)	$dproj_{t_{dmin}-90}$ (m)	$dreal_{t_{dmin}-100}$ (m)	$dproj_{t_{dmin}-100}$ (m)
2307	3000	51,298	453.6	0	453.8	453.6	...	1123.9	453.7	1229.4	453.7
2307	3005	51,447	400.5	0	400.6	400.5	...	1435.4	400.5	1583.8	400.5
2307	3012	51,223	534.1	0	534.6	534.1	...	1579.9	534.1	1735.6	534.1
2307	3037	51,527	240.6	0	241.1	240.6	...	1272.3	240.6	1408.7	240.6
2931	2946	51,974	1.3	1	19.9	0.4	...	1629.5	650.9	1729.7	91.4

¹ Instant of time of minimum distance between the UAS pair. ² Minimum distance between the UAS pair. ³ Real distance at $t_{dmin}-N$ seconds. ⁴ Projected distance at $t_{dmin}-N$ seconds.

In order to evaluate the correlation between the minimum distance at the closest point of approach ($dmin$) for each pair of UAS and the real and projected distances N seconds before that instant of time, we calculated the Pearson correlation coefficient:

$$corr_{dmin-dist_N} = \frac{cov(dmin, dist_N)}{\sigma_{dmin} * \sigma_{dist_N}} \quad (6)$$

Figure 7 shows the correlation with $dmin$ of the real and projected distances N seconds before this point. It can be observed that up to 10 s before the minimum distance timespot, there was a greater correlation between the real distance N seconds before this timespot and the minimum distance at the closest point of approach of UAS pairs, which was above 99% up to 5 s in advance. However, at earlier times, the correlation of the projected minimum distance was greater than that of the real distance; this effect could be counterintuitive, but it indicates that the projected minimum distances were more stable in time, whereas the real distances could increase in different proportions, depending on the variations in bearings.

Correlation of distances measured and projected with minimum distance

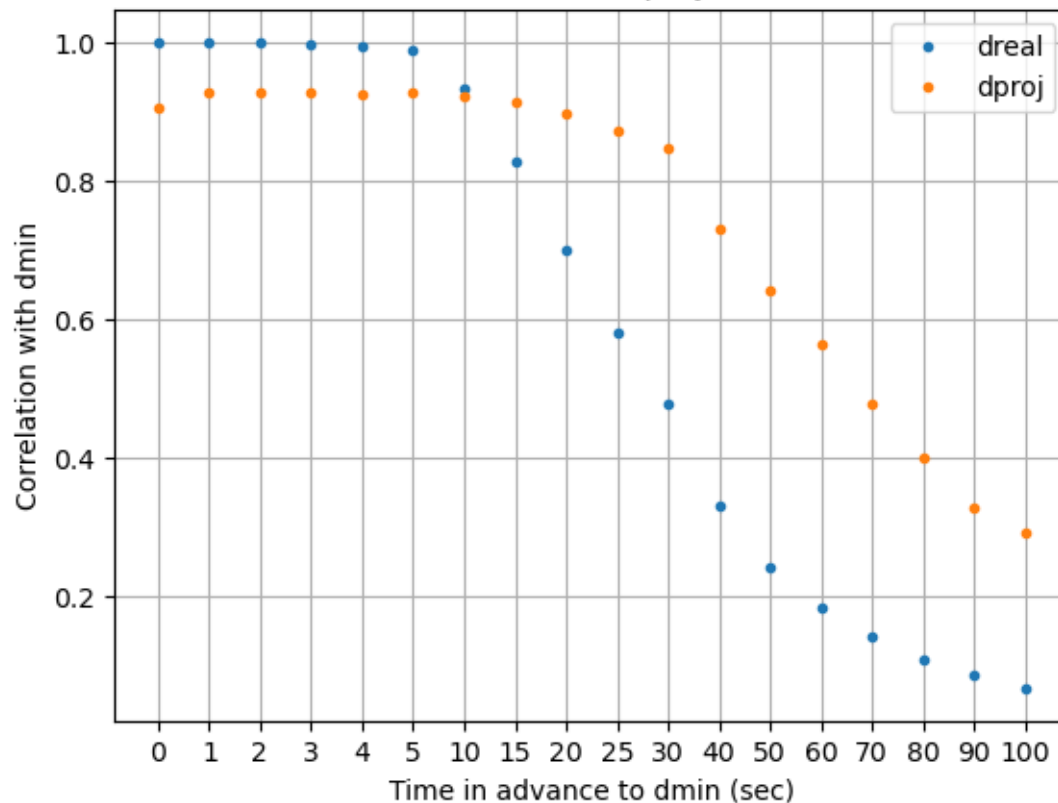


Figure 7. Correlation of real and projected distance before the closest point of approach with the minimum distance.

2.1. Machine Learning Algorithms

The objective of this research was to evaluate the ability of machine learning (ML) algorithms to predict near collisions between UAS trajectories during the flight. The dataset used presented pairs of UAS, providing the minimum distance between them and classifying observations as a near-collision when this minimum distance (d_{min}) was lower than a predefined threshold, which, in this paper, was set at 50 m. This classification was the dependent or categorical variable in our dataset. In addition, the dataset provided the real distance between each UAS pair and the minimum projected distance several moments of time before the instant of the closest point of approach; these data were used as the independent variables in the model.

Taking into account the fact that we have labelled data and that we intend to classify new data based on the training obtained with previous example pairs (train data), this is a case of supervised learning. Furthermore, since we sought to predict discrete categories (near-collision, no near-collision), this was a classification problem and, in particular, a binary classification. The most popular ML algorithms for binary classification [33,34] are:

- Logistic Regression;
- K-Nearest Neighbors (KNN);
- Support Vector Machine (SVM);
- Artificial Neural Network (ANN);
- Decision Tree—Random Forest.

Additionally, a linear regression model was also tested. Linear regression predicts a dependent variable based on coefficients multiplying the independent variables, calculated

to minimize the residual sum of squares between the actual value (Y) in the train dataset and the predicted value (Y_{pred}).

$$Y_{pred} = \alpha_0 + \alpha_1 \cdot X_1 + \alpha_2 \cdot X_2 + \dots + \alpha_n \cdot X_n \quad (7)$$

However, linear regression provides continuous values, not categorical values; therefore, in this case, the numerical prediction obtained was then classified with the same criteria as the minimum distance, i.e., labelled as a near collision if the minimum distance predicted was lower than 50 m.

Logistic regression is based on linear regression but adapted for the case in which the dependent variable is categorical, not continuous. Logistic regression transforms the numeric value of the linear regression into a probability using the sigmoid function.

$$Prob(Y = 1) = \frac{e^{(\alpha_0 + \alpha_1 \cdot X_1 + \alpha_2 \cdot X_2 + \dots + \alpha_n \cdot X_n)}}{1 + e^{(\alpha_0 + \alpha_1 \cdot X_1 + \alpha_2 \cdot X_2 + \dots + \alpha_n \cdot X_n)}} \quad (8)$$

The sigmoid function presents values very close to either 0 or 1 through most of its domain. Then, if the predicted output is greater than 0.5 for a certain observation, it is classified as 1, and otherwise as 0.

K-Nearest Neighbors (KNN) is a method for classifying based on the similarity to other cases, based on the idea that similar cases with the same class labels are near each other. To calculate the similarity considering several independent variables, the algorithm evaluates the root square of the sum of the Euclidean distances for each independent variable. Then, to choose the number of neighbors (K), the most adequate method is to evaluate the accuracy for different K values with the training data and apply this K value to the test data.

Support Vector Machines (SVM) groups data based on the category in which they are classified, even if the data are not linearly separable, i.e., by means of a curve in a high-dimensional feature space. Then, the data are transformed so that this curve becomes a hyperplane, which acts as a separator among categories, classifying the new based on the side of the hyperplane in which they fall, once transformed.

Artificial Neural Networks (ANN) emulate biological neural networks, defining nodes connected by weighted links, based on the relevance of their outcome for the training data; the form of the relationships is established during the learning process.

A Random Forest is an ensemble technique, i.e., it combines the predictions of multiple decision trees to create a more robust prediction. In a random forest, the growth of the decision trees is determined by random vectors and the combined result is obtained by averaging or, in the case of classification, by choosing the most frequent result. As the number of trees in the forest grows, the generalization error converges to a limit.

The ML algorithms tested are the most common for binary classification and, therefore, they have been considered the most adequate to begin the research in the topic explored by this paper. However, other algorithms are more adequate for imbalanced classification, such as SMOTE [35], so they will be tested in future research.

2.2. Evaluation Metrics

Having defined and implemented the ML models, it was necessary to evaluate to what extent they were able to predict near-conflicts for new observations. The most common metrics to analyze ML models performance [36,37] are:

- Jaccard Index;
- Accuracy;
- Confusion Matrix;
- Precision;
- Recall;
- F1-score;
- Precision-Recall Area Under the Curve (PR AUC).

The Jaccard Index or similarity coefficient is the ratio between the number of correctly predicted values and the total number of values plus those incorrectly predicted. In other terms, it can be expressed as the intersection of the actual values (Y) and the predicted values (Y_{pred}), divided by the union of both of them.

$$J(Y, Y_{pred}) = \frac{|Y \cap Y_{pred}|}{|Y \cup Y_{pred}|} = \frac{|Y \cap Y_{pred}|}{|Y| + |Y_{pred}| - |Y \cap Y_{pred}|} \quad (9)$$

The Accuracy Index is the ratio between the number of correctly predicted values and the total number of values:

$$Acc(Y, Y_{pred}) = \frac{|Y \cap Y_{pred}|}{|Y|} \quad (10)$$

Both metrics are common in classification problems but can be non-representative in imbalanced classifications. For example, in this case, in which we have a dataset of 13,869 pairs, from which only 1335 pairs are classified as near-collisions, let us imagine that our model predicts all the observations as non-near-collisions. This would be a very bad model, as it would be totally unable to predict near-collisions. However, the Jaccard Index and the accuracy values would be quite high:

$$J(Y, Y_{pred}) = \frac{13,869 - 1335}{13,869 + 1335} = 0.824$$

$$Acc(Y, Y_{pred}) = \frac{13,869 - 1335}{13,869} = 0.904$$

Therefore, they will be disregarded in the evaluation of the results of this article.

The Confusion Matrix is a table comparing the different combinations of real and predicted values, identifying true positives and negatives, as well as false ones, thereby providing a full view of the capability of the model to predict the real classes and the errors made. In a perfect model, the numbers out of the main diagonal would be zero. Table 2 represents the different elements of a confusion matrix.

Table 2. Confusion Matrix metric.

		Predicted Values		
		No Near-Collision	Near-Collision	
Actual Values	No Near-Collision	True Negative	False Positive (False Alarm)	$Precision = TP / (TP + FP)$
	Near-Collision	False Negative (Missed Detection)	True Positive	
$Recall = TP / (TP + FN)$				

The confusion matrix is really useful to understand the accuracy of the model, but it is not really a metric, so it cannot be easily compared. In our case, the most relevant values are false negatives, i.e., the real near-collisions which the model has not detected (missed detections); this value should tend to zero in a reliable and safe model. Also relevant are false positives, i.e., the real non-near-collisions which are predicted as near-collisions. These errors would not be as critical as false negatives, but they should also be as low as possible, because they would lead to false alarms, affecting trajectories that are not really a problem.

Recall and Precision are metrics generated from the results of the confusion matrix results and are used to evaluate and compare models.

Recall tells us about the proportion of real near-collisions predicted by the model, i.e., the ratio of the true positives to the total number of real near-collisions. This is one of the most relevant metrics in this research, as it informs about the rate of missed detections.

Precision, in contrast, is the proportion near-collisions predicted by the model that are true, i.e., the ratio of the true positives to the number of predicted near-collisions. This is also an important metric, as it informs about the rate of false alarms.

Recall and Precision are very representative metrics in our case. For example, in the case explained earlier where our model predicted all the observations to be non near-collisions, both Recall and Precision would be zero. In the opposite case, i.e., where the model would predict all the observations to be real collisions, the Recall value would be 1 but the Precision would be zero.

In order to evaluate both metrics simultaneously, the *F1*-score provides the harmonic mean of the precision and recall, which can be interpreted as a weighted average of both metrics, biased to the lower of them.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (11)$$

Finally, we will measure the Precision-Recall Area Under the Curve (PR AUC), which has been demonstrated to be more appropriate for imbalance domains due to its focus on class-specific performance [38]. The precision-recall curve plots recall values on the *x*-axis and precision values on the *y*-axis. In a perfect model, both precision and recall would be 1, and the curve would cover the entire plot, with the AUC also being 1. The curve is usually non-linear, as precision has an inverse relationship with recall. Figure 8 presents an example of such a curve.

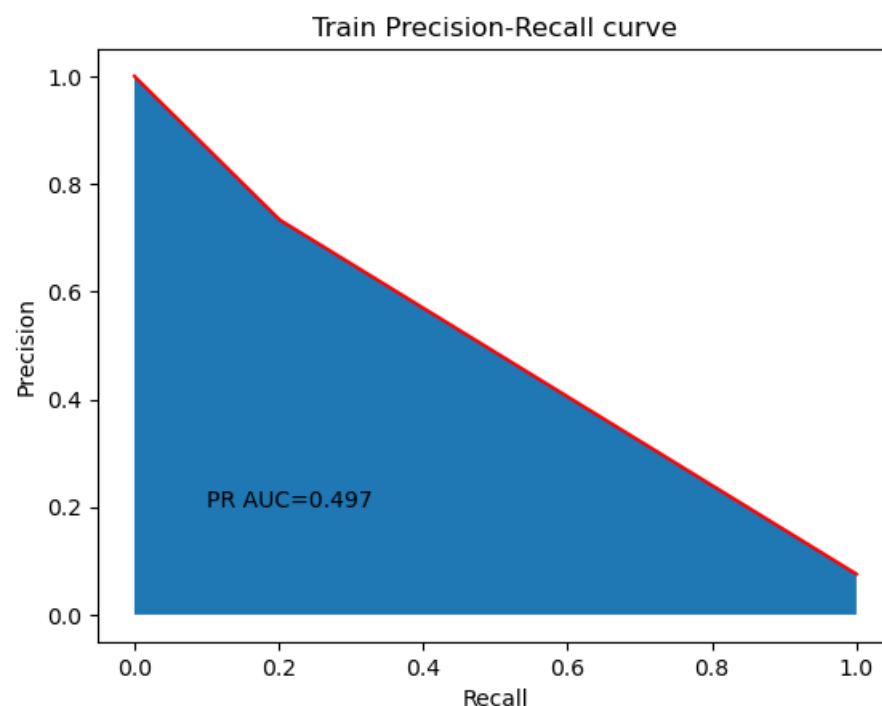


Figure 8. Precision-Recall Area Under the Curve example.

The PR curve calculated different values for precision and recall, using different thresholds. Therefore, when comparing different models, this curve allowed us to evaluate not only which model performed better, but also the impact of different thresholds for defining classes.

3. Results

This section presents the results obtained from the implementation of the ML algorithms described in Section 2.1 for the dataset of trajectories considered in this article. All the models were developed in Python 3 using the scikit-learn library [39].

The algorithm with the highest accuracy, among those defined in Section 2.1, was the Random Forest classifier. For the sake of brevity, only the results for this algorithm are presented in this section; the results for the rest of the algorithms can be found in Appendix A.

The dataset described in Table 1 was split in two parts, with 80% of the samples for training and 20% for testing the models.

Three models were developed for each algorithm:

- one based only on real distances N seconds before the closest point of approach and all the previous distances measured, up to 100 s before the minimum distance time;
- another one based only on the minimum projected distances N seconds before the closest point of approach and the previous minimum projected distances;
- finally, a model based of both the distances measured and the minimum projected distances, considering all the data from N seconds to 100 seconds before the closest point of approach.

The three models for each algorithm were calculated repetitively for each value of N seconds before the closest point of approach contained in the dataset, i.e., 1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 40, 50, 60, 70, 80, 90, and 100 s. Obviously, the lower the time to the closest point of approach, the greater the number of data considered in the model. For example, for 1 s, 17 distance values were available for each observation; for 30 s, only 8 distance values and, for 100 s, only one distance datum was provided to the model.

As the distances tended to increase with the time toward the closest point of approach, the models considered the standard score, i.e., the distances for each timespot were normalized by removing the mean and dividing by the standard deviation.

To easily see how these models work, we implemented a multiple linear regression (MLR) to analyze the correlation of the minimum distance with the set of distances considered in the three models considered for each algorithm. Figure 9 shows the scatter plots of the real minimum distance versus the predicted minimum distance 5 s before the closest point of approach for the three models implemented in this algorithm. The figures show that the regression based on real distances was much more accurate than the one based on projected distances.

To numerically evaluate the correlation, we had to measure the accuracy of a regression model. The most popular metric to measure accuracy is the coefficient of determination, or the R^2 score:

$$R^2 = 1 - \frac{\sum (Y - Y_{pred})^2}{\sum (Y - \bar{Y})^2} \quad (12)$$

Figure 10 presents the R^2 score evolution with time toward the minimum distance timespot, comparing the three models and also the R^2 score calculated for the projected distance at each previous moment of time considered.

The results, considering the MLR models show that the prediction based on the previous distances performed better than the prediction based on the projected minimum distances up to 40 s before the closest point of approach. From that time onward, the MLR based on previous projected distances performed slightly better than the one based on real distances, although both had very poor correlations (R^2 close to zero). But the regression based on both distances, real and projected, performed better than the models based on only one of those groups of variables, making it the best predictor.

Summarizing the results for the combined model, we see that the predicted minimum distance was quite accurate up to 15 s before the minimum distance timespot and reasonably accurate up to 25 s in advance.

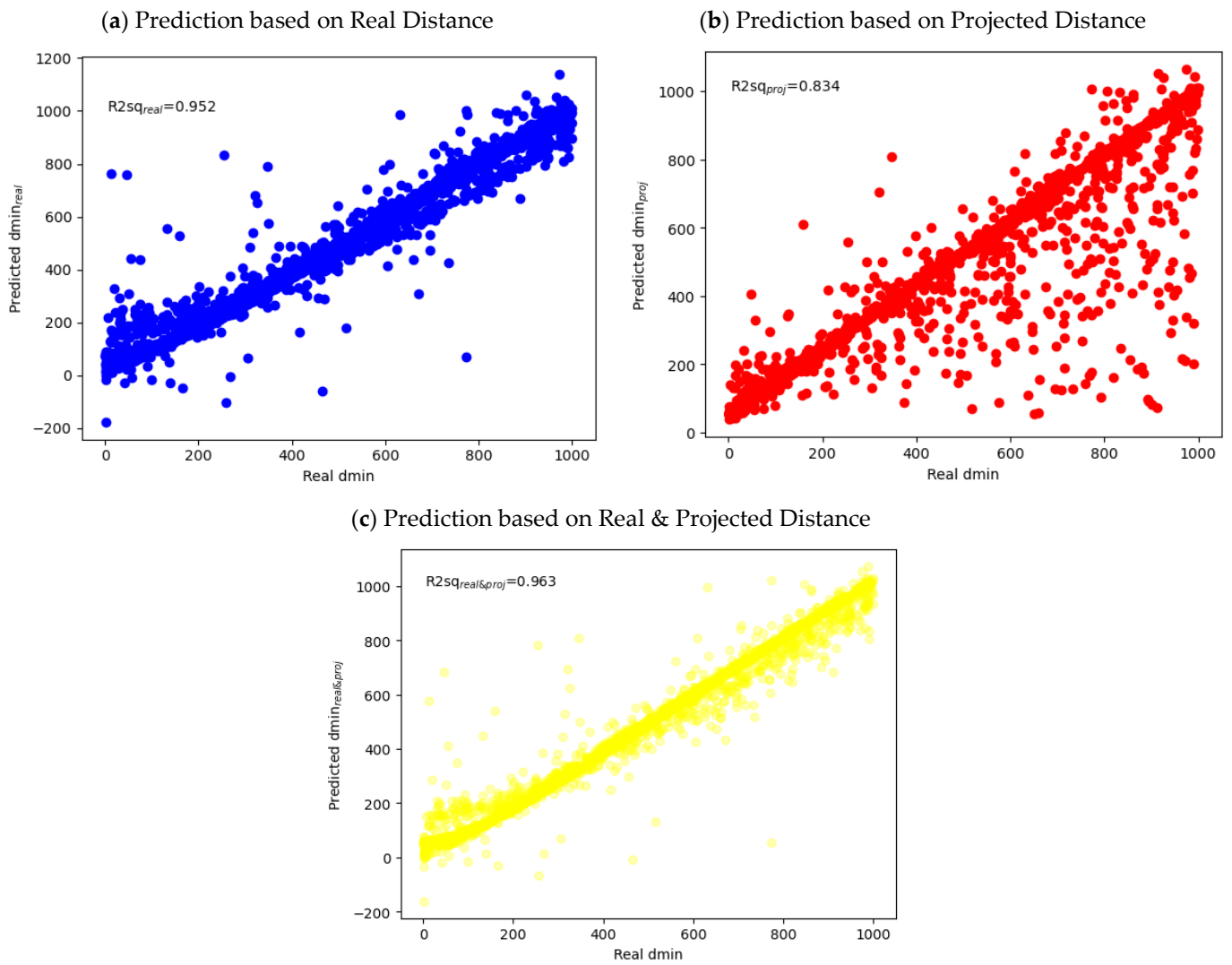


Figure 9. MLR Scatter plots: predicted minimum distances vs. real minimum distance 5 s before the closest point of approach.

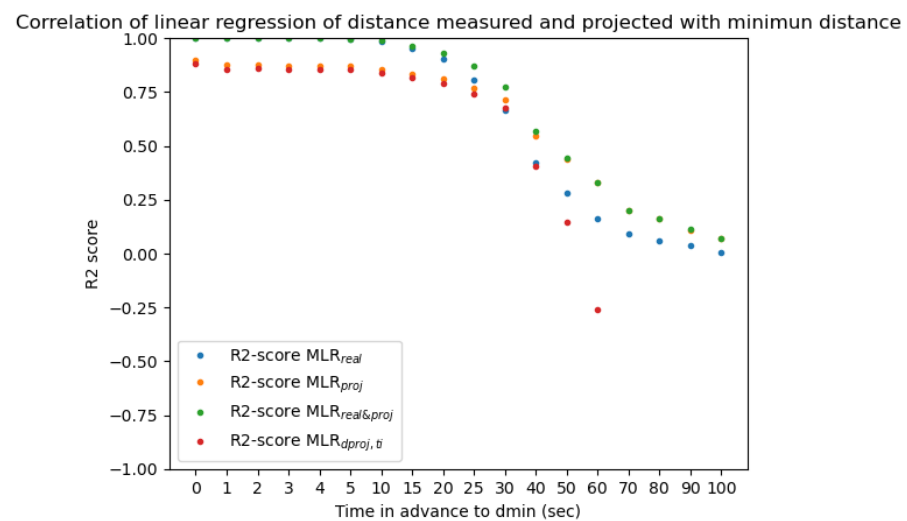


Figure 10. R2 score for the three MLR models.

Additionally, we can see that the correlation of the projected minimum distance (d_{proj}) at each timespot always performed worse than the multiple linear regressions of the measured and projected distances. In fact, the prediction based on the projected distance at time $t_{dmin}-T$ was worse than the average distance (R^2 score < 0) as a prediction 60 s in advance. As explained, the projected minimum distance at each timespot represents the minimum distance that a pair of UASs would reach if they continued flying with the same bearing and speed as they had at that moment. This is the most accurate analytical method developed to determine potential conflicts among pairs of aircraft. Therefore, these initial results show that even the simplest ML algorithm, i.e., a linear regression, provides a greater correlation with the real minimum distance than the correlation that would be obtained analytically by projecting the speed vectors of each pair of UAS to calculate the minimum potential distance.

However, these results can be misleading, as a good R^2 score in a regression does not imply that a MLR model can correctly classify near-collisions, i.e., minimum distances lower than 50 m for a pair of UAS in this article. The R^2 score is an average result of the residuals, i.e., the differences between the real minimum distance value and the predicted one. Therefore, the resulting average may be close to zero, balancing positive and negative residuals, which is what is happening in this case, as we can see in Figure 9c.

In order to assess whether the MLR model accurately predicted near-collisions, it was necessary to classify the results (1 for distances lower than 50 m and 0 for greater minimum distances) and evaluate them with respect to the real minimum distances. The results of this evaluation can be seen in Appendix A. Additionally, Figure 11 presents the evolution of the $F1$ -score results of the three MLR models with the time to the closest point of approach, together with the results that would be obtained for a classification made for the real distances and projected distances at each timespot (i.e., the real and projected distances at each time in advance were classified as 1 or 0, if their values were greater or lower than 50 m; then, they were compared with the minimum distances classification).

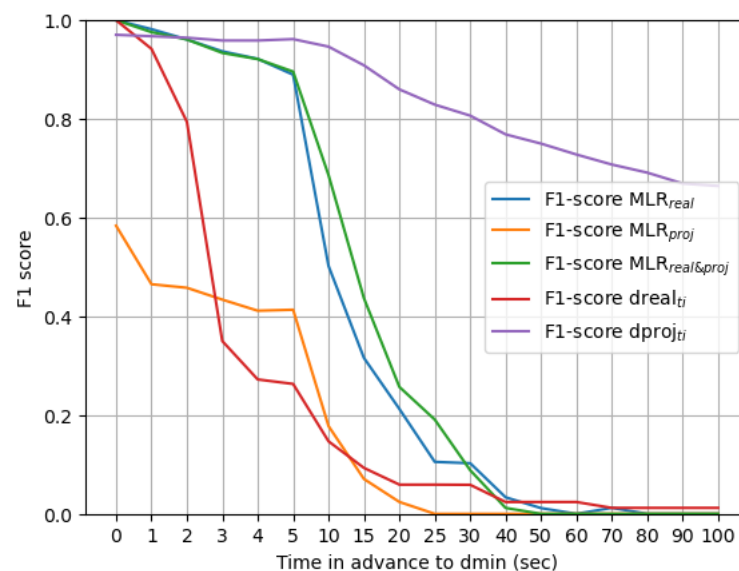


Figure 11. $F1$ -score classification results for MLR models, real and projected distances.

The obtained results show that the classification based on MLR regression only performed reasonably well up to 5 s in advance before the closest point of approach and only for the models that included the real distance. A comparison with the classification based on real distance was a worst predictor than the MLR models but, in contrast to the results of the correlation shown in Figure 10, the classification based on the projected distance at each timespot was a better predictor than the MLR models. To explain the differences between the results of the correlation based on the R^2 score, Figure 12 shows the scatter

plots of the real minimum distance versus the projected minimum distance 5 s before the closest point of approach. It can be seen that the projected distance was generally lower than the minimum distance, so the errors were biased toward the lower part of the graphic, which is the reason why the R^2 score values were worse than for the MLR models but the $F1$ -score classification values were significantly better. Therefore, in the analyzed scenario, the projected distance at each timespot was a better predictor of conflicts than the MLR models. However, it is important to remember that this scenario included only direct point-to-point flights, as shown in Figure 2, and did not include other flight patterns, such as scanning flights.

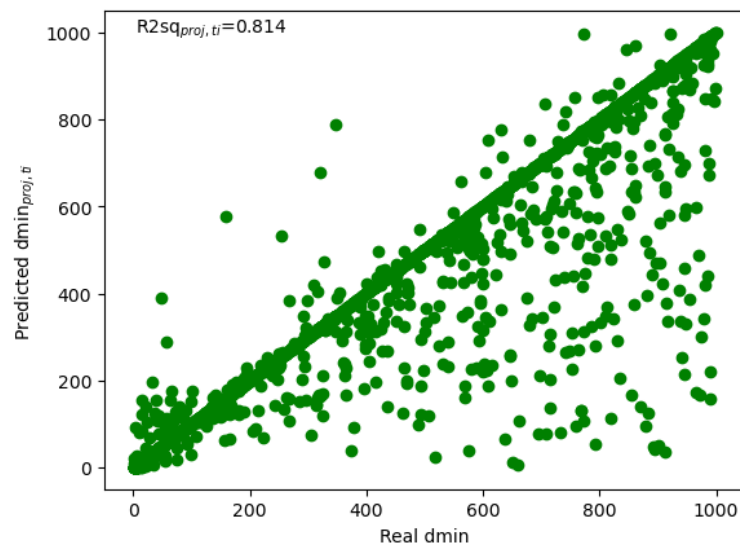


Figure 12. Scatter plot: real minimum distances vs. projected minimum distance 5 s before the closest point of approach.

3.1. Random Forest

The most accurate algorithm implemented was the random forest classifier. This algorithm creates a set of decision trees from a randomly selected subset of the training set and ensembles them, i.e., combines their predictions, averaging the results to improve prediction accuracy and control overfitting.

As this is a very complex model, where many hyperparameters need to be tuned to improve its accuracy, in this model, we applied the GridSearchCV tool. Table 3 shows the hyperparameters considered in the optimization and the ranges applied; additional parameters such as 'bootstrap', 'max_features' or 'min_sample_split' had to be disregarded due to the computational cost. The optimal values identified for tuning depend on the time in advance to the closest point of approach and the type of distances considered.

Table 3. Random Forest Hyperparameter search.

Parameter	Meaning	Range Considered in the Optimization
max_depth	Maximum depth of the decision tree	5, 10, 15, 20, 40
n_estimators	Number of decision trees	100, 200, 300, 500, 1000
min_samples_leaf	Minimum number of samples required to be at a leaf node	1, 2, 4

The algorithm was implemented for the three models defined with the optimum hyperparameters identified in the search. Figure 13 shows the results for each one.

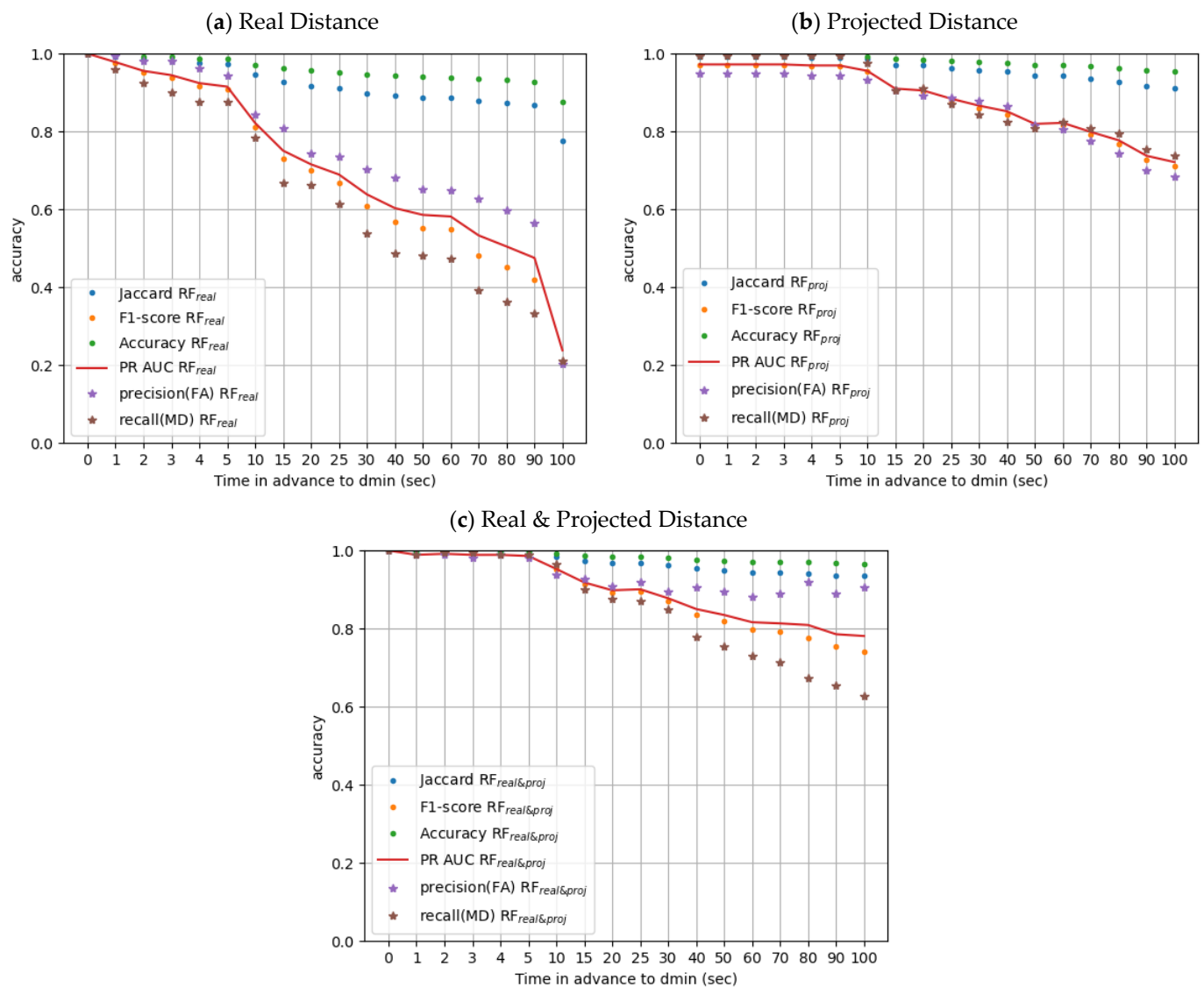


Figure 13. Random Forest Models Evaluation Metric Results.

In this case, the model based on real distances only performed better than the one based on projected distances for 1 s before the minimum distance timespot. The model combining both real and projected distances performed better than both of them. Figure 14 compares the results of the three models. In the combined model, the precision, recall and F1-score values were higher than 0.985 up to 5 s in advance before the instant of the closest point of approach. These values represent missed detections and false alarms tending to zero. The recall and F1-score remained above 0.95 up to 10 s, which means that the majority of the near-collisions were detected with a sufficient reaction time even for human pilots [40]. From that timespot, the precision value remained above 0.9, but the recall value deteriorated progressively to 0.6 at 100 s in advance. However, the F1-score value remained quite high for all the time analyzed.

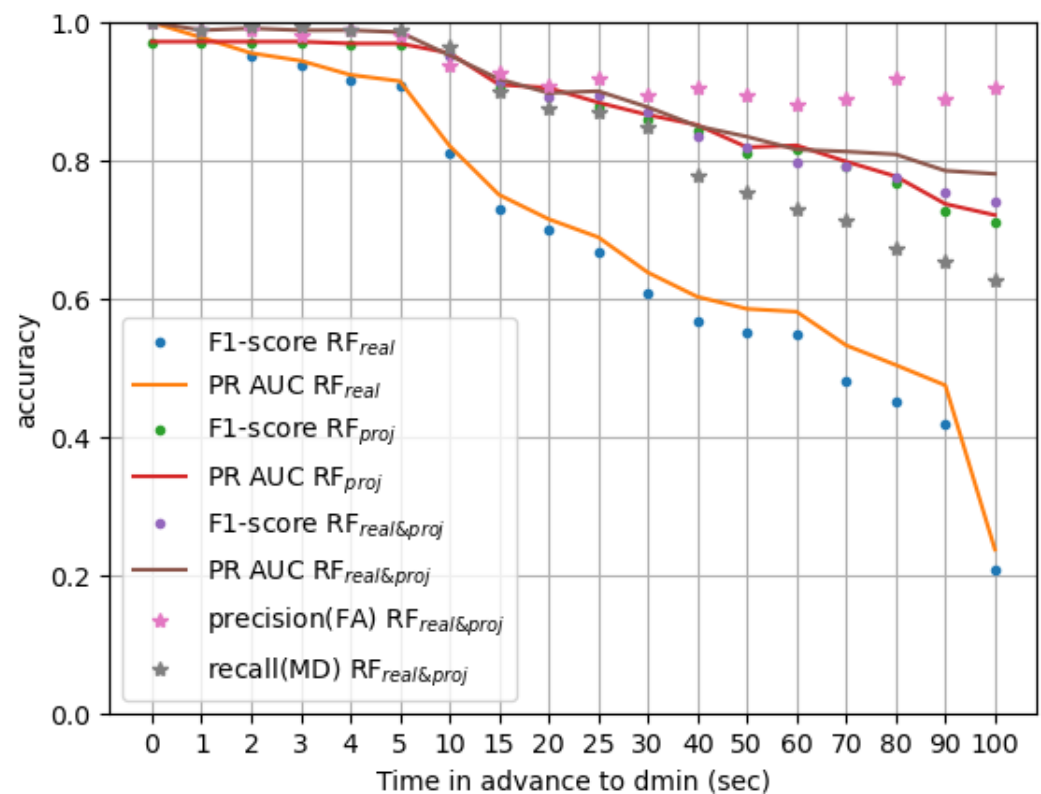


Figure 14. Random Forest models F1-score and PR AUC comparison.

3.2. Summary of ML Algorithms Results

Although random forest provided the most accurate results, artificial neural networks and logistic regression provided similar results for long and short times before the closest point of approach, respectively.

Figure 15 shows the recall (missed detections), precision (false alarms) and F1-score for these three algorithms. Additionally, it shows the results obtained with the analytical classification based on the projected distance at each timespot, which provides worse classification metrics than the ML models, especially when compared with random forest. These graphs also show that random forest and logistic regression would be able to detect almost all conflicts up to 5 s before they occurred, generating almost no false positives (false alarm). The recall parameter remained higher than 0.95 for the three algorithms up to 10 s, which should be a sufficient reaction time to modify the trajectories of the UAS once a conflict has been detected, according to [41], that defines 3 s for autonomous UAS and 7 s for piloted ones. Therefore, these ML algorithms could be safely applied to prevent UAS collisions during flight, without prior strategic deconfliction.

The random forest model would maintain a recall rate above 0.8 up to 40 s before the closest point of approach, while the precision rate would be above 0.9 up to 100 s in advance. The other two algorithms, particularly ANN, also presented better values of precision versus recall. This was an undesired result, as detecting near-collisions (safety) should have priority over minimizing false alarms (efficiency). This problem could eventually be solved by prioritizing recall, with appropriate tuning of the model, which will also be explored in future work.

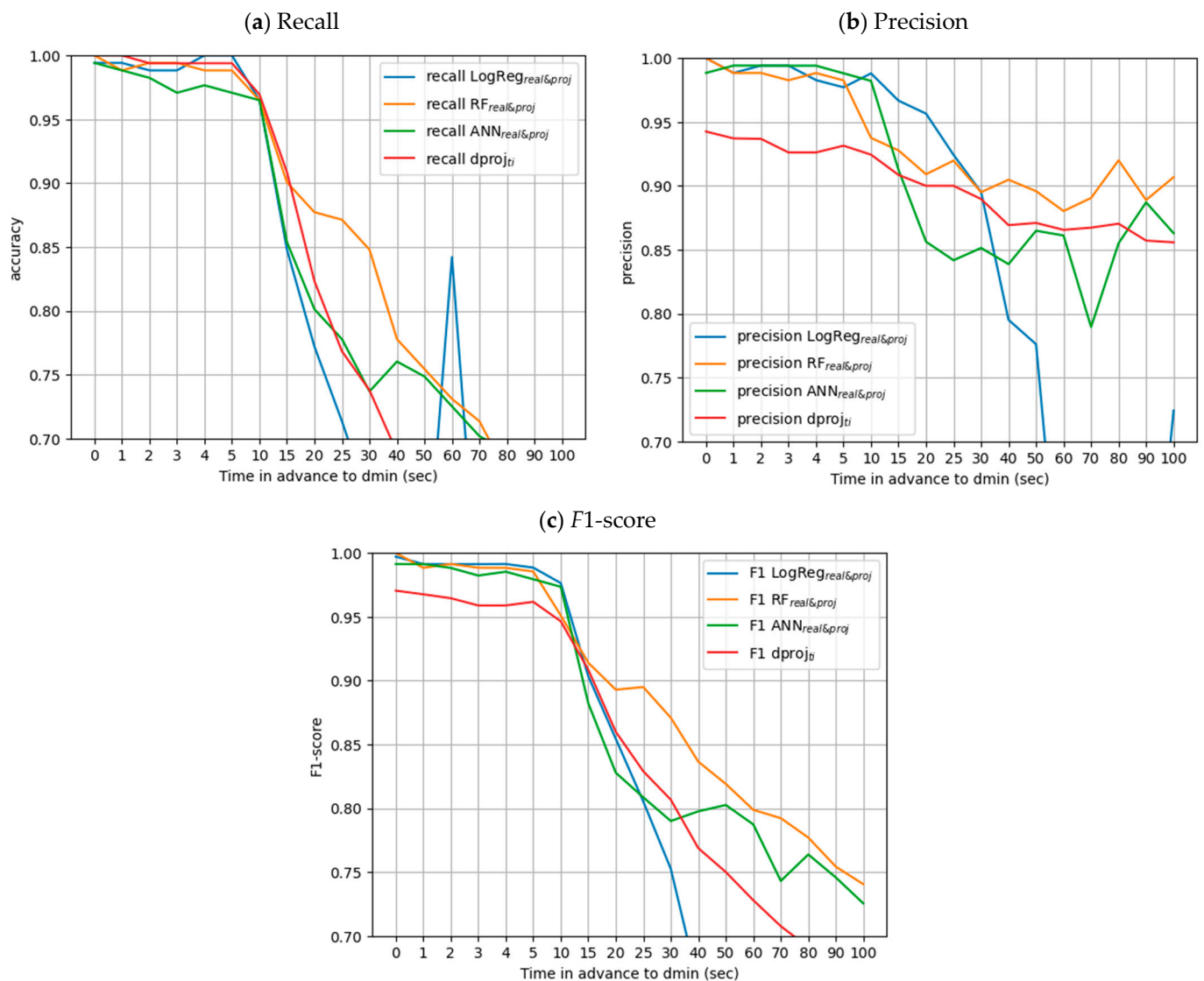


Figure 15. Best ML models comparison with analytical technique.

3.3. Sensitivity to the Near-Collision Distance Threshold

As explained above, this paper classifies the minimum distance of a UAS pair as a near-collision if their distance from one another was less than 50 m. However, this threshold is tentative, and different ones could be applied. This section analyzes the impact of the threshold distance on the accuracy provided by the random forest algorithm. For this purpose, the results obtained for the reference threshold of 50 m were compared with the ones obtained for 100 m and 25 m. Figure 16 shows the recall and false negatives (missed detections), precision and false positives (false alarms), and the F1-score for these three thresholds using the random forest algorithm for the real and projected distance.

The results show that up to 30 to 40 s in advance of the closest point of approach, the models performed better with a larger distance threshold for near-collisions. In fact, the 100 m threshold did not generate any missed detections up to 5 s from the minimum distance timespot. On the contrary, from 15 s in advance, the model based on the smallest threshold generated a lower number of false negatives (missed detections), and from 30–40 s, it performed better globally than the other two models. The reference model, which considered a threshold of 50 m, seemed to present a good balance between the results of the other two models.

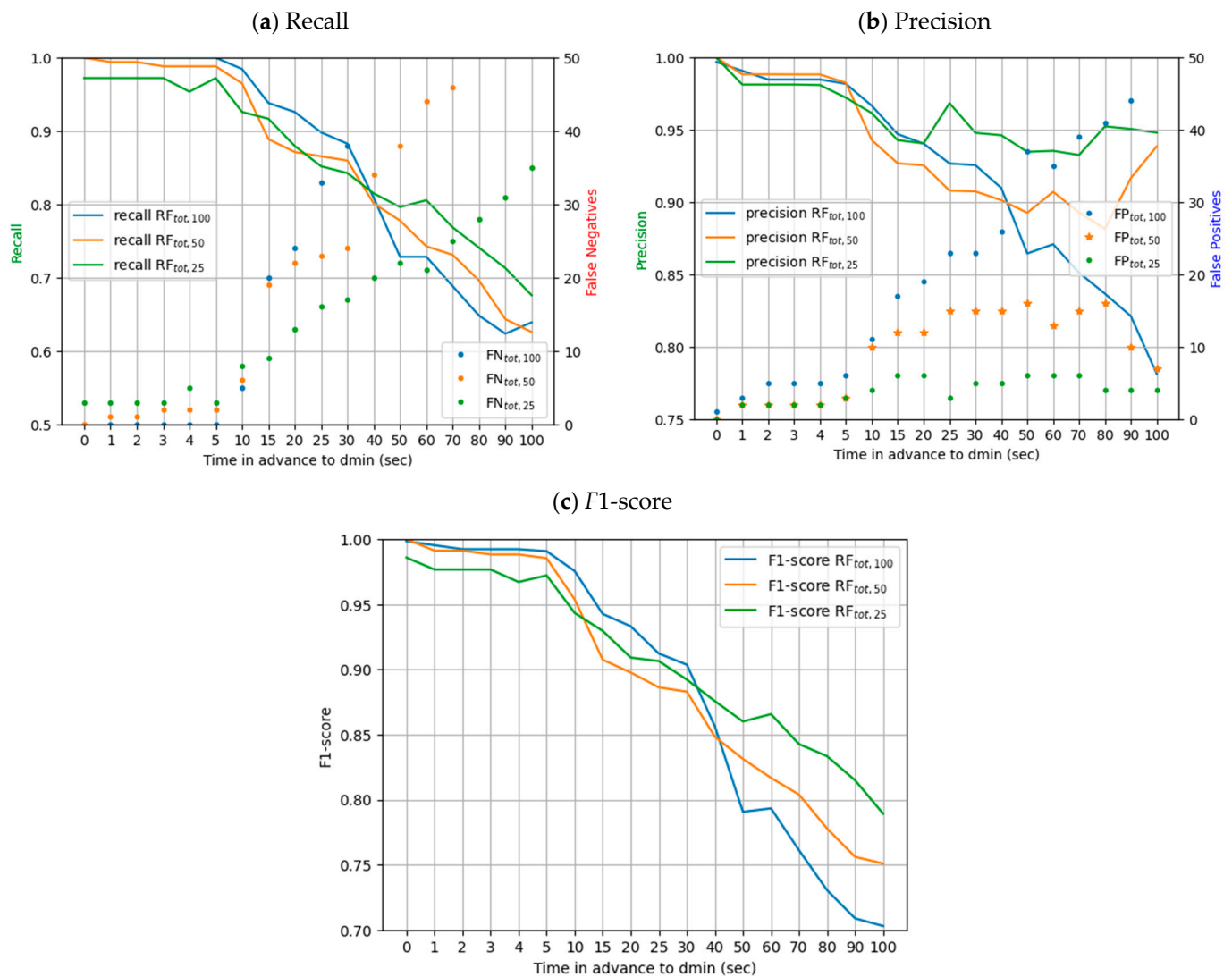


Figure 16. Random Forest results for different near-collision distance thresholds.

3.4. Adequacy of the Dataset Size

One of the important issues in ML is to gather big enough datasets to train the models in order to guarantee the consistency of the obtained results. To evaluate the adequacy of the size of the dataset considered in this paper, Figure 17 presents a learning curve which shows cross-validated training and test scores for different training set sizes. In this figure, we can see how the *F1*-scores of the training and test data converged with increasing percentages of the dataset used.

The figure shows that there was a significant convergence as the dataset grew, with the difference almost becoming stabilized for the total size considered in this paper.

Additionally, Figure 18 presents the *F1*-scores obtained for the random forest algorithm, training it with 25%, 50%, 75%, and 100% of the dataset considered in this paper and keeping the same amount of test data. It can be seen that the results tended to converge, becoming fully stable for the 100% dataset.

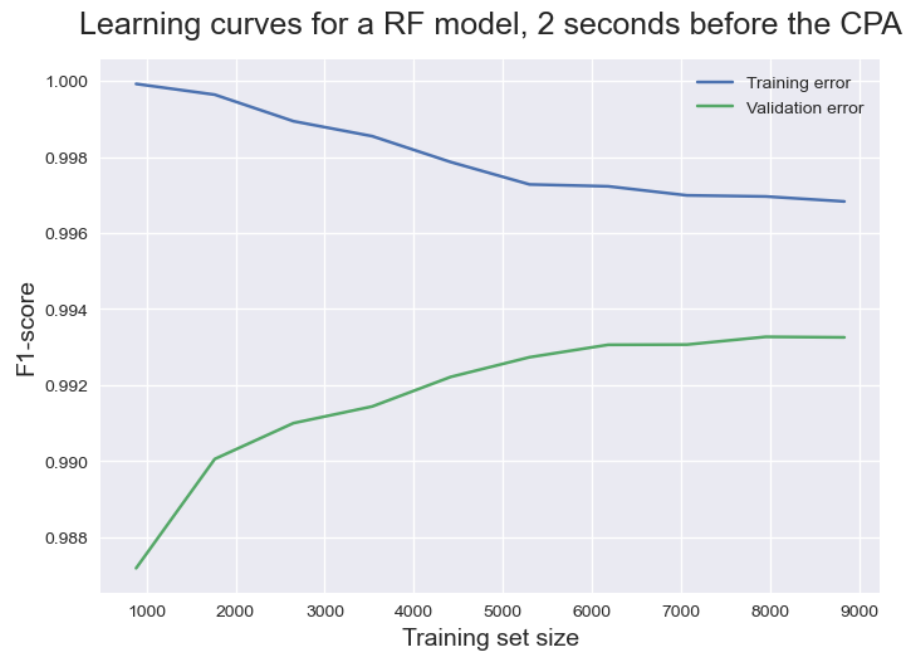


Figure 17. Learning curve for the dataset analyzed.

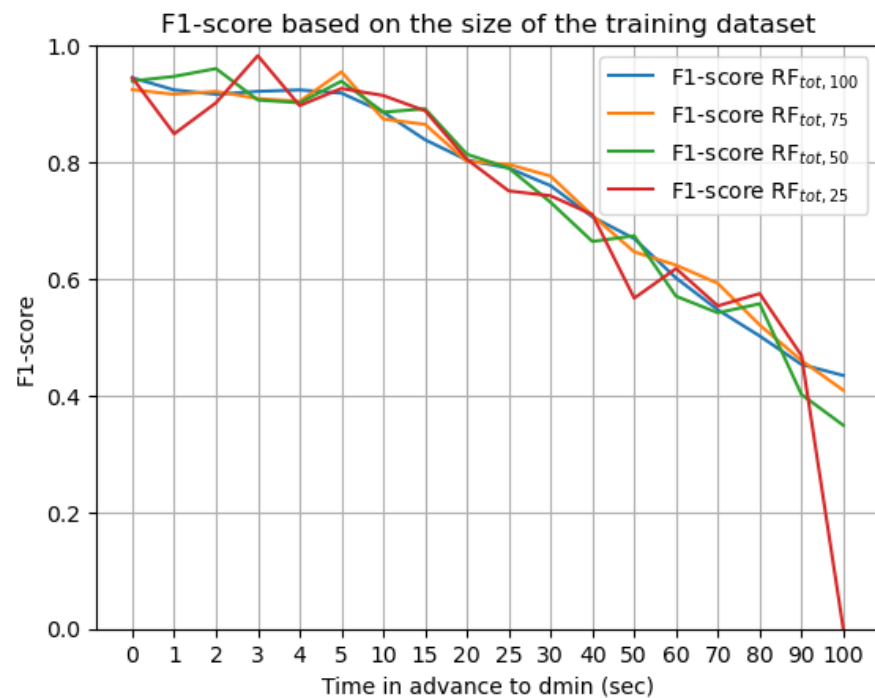


Figure 18. Variation of the F1-score for the RF model with different dataset sizes.

4. Conclusions

The results of this research suggest that machine learning algorithms could potentially be applied to identify conflicts during the flight, preventing UAS collisions.

Three models were implemented for each algorithm: one based on real distances only, one based on projected minimum distances only and, finally, one combining both real and projected distances. In general, the last model provided the best performance for each algorithm; the obtained results indicated that ML models worked better when different pieces of information were taken into account. In the future, it would be interesting to include other parameters in the models, such as relative bearings and speeds between UAS pairs, in order to improve the predictions.

As expected, some models provided much better performance than others, with random forest, artificial neural networks and logistic regression being the most accurate algorithms, while the other algorithms tested did not seem to be adequate to detect potential near-collisions, providing worse results even than a classification based on an analytical technique such as the projected minimum distance, calculated from the bearings and speeds of the UAS at each moment of time.

The good results of these three techniques indicate that all of them, and especially random forest, could be applied to prevent conflicts, even though the obtained results did not allow us to determine whether the pairs of UASs which became false positives and negatives were the same for the three models or whether they were different for each model. In the latter case, the simultaneous application of several ML techniques would allow conflicts to be detected earlier and more accurately. In any case, the use of more than one algorithm seems to be a good practice to provide more robust conflict detection results. Additionally, future research will explore more sophisticated ML algorithms to deal with imbalanced datasets in order to compare the results with those obtained with the most common ML algorithms for binary classification.

It is also important to note that the ML obtained results are applicable in the same or similar scenarios, i.e., where the UASs and type of operations performed are similar to other previous operations. Otherwise, it is unlikely that the predictions made could be applicable.

Disregarding a previous strategic deconfliction is a major challenge, and probably a minimum identification of the most conflictive flight plans will be required. However, the strategic deconfliction services defined in the current standards and regulations were not able to handle the large number of operations expected in the future. Future work will explore how tactical deconfliction based on ML algorithms could perform after prior strategic deconfliction of the most critical UAS pair flight plans.

Another interesting issue is the impact of the distance threshold defined to classify near-collisions on the accuracy of the models. Larger thresholds provided better results when the time to the closest point of approach was small, which would be expected behavior, as a larger threshold facilitated classification and provided a more balanced dataset. However, it can be considered surprising that, for larger times to the closest point of approach, the models developed for a smaller threshold performed much better and generated much lower numbers of missed detections. These results suggest that it is advisable to use models based on smaller thresholds when a UAS pair appears to be far away from its closest point of approach, and to combine these with models based on larger thresholds when they approach the minimum distance timespot.

The weakest point of the analysis was that the algorithms were fed with a processed dataset, in which the columns present the distances 1, 2, 3, . . . N seconds before the closest point of approach for every UAS pair. In this way, the data were homogeneous, and the ML models could be adequately fitted. However, in reality, each UAS trajectory would take place at a different moment in time, and the closest points of approach for each UAS pair would not be synchronized. Therefore, it remains a challenge to test whether the developed algorithms will still work efficiently when applied to asynchronous UAS data. To that end, it will be useful to include in predictions not only if a conflict will occur, but also when it will take place. Future work will also consist of implementing ML regression techniques to identify the time to the closest point of approach for each UAS pair.

Author Contributions: Conceptualization, V.G., J.A.P.-C., L.P.S., Y.X. and L.S.-M.; methodology, V.G.; software, V.G. and J.A.P.-C.; validation, V.G., J.A.P.-C., L.P.S., Y.X. and L.S.-M.; formal analysis, Y.X., J.A.P.-C. and L.P.S.; data curation, V.G.; writing—original draft preparation, V.G.; writing—review and editing, V.G., J.A.P.-C., L.P.S., Y.X. and L.S.-M. All authors have read and agreed to the published version of the manuscript.

Funding: No funding has been applied to this research.

Data Availability Statement: The traffic data samples and the results shown in Sections 2 and 3 can be found in https://github.com/vgordo/ML_UAS_Tactical (accessed on 25 September 2024).

Acknowledgments: This paper uses data provided by the DACUS consortium. We want to thank them for their contribution.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A

This Appendix presents the detailed results of the rest of ML algorithms described in Section 2.1 but whose results are not presented in Section 3, for the sake of brevity.

Appendix A.1. Linear Regression

The first model implemented is a Multiple Linear Regression (MLR), i.e., a regression model, not a classification one. This model has been implemented to test the accuracy of a regression, as described in Section 3 and then, to evaluate the goodness of these MLR models to predict Near-Collisions, we have developed an identical classification of the minimum distances predicted by the regression models.

This classification (1 for distances lower than 50 m and 0 for greater minimum distances) has been evaluated against the real minimum distances' classification, using the metrics defined in Section 2.2. Figure A1 presents the results for the three MLR models developed.

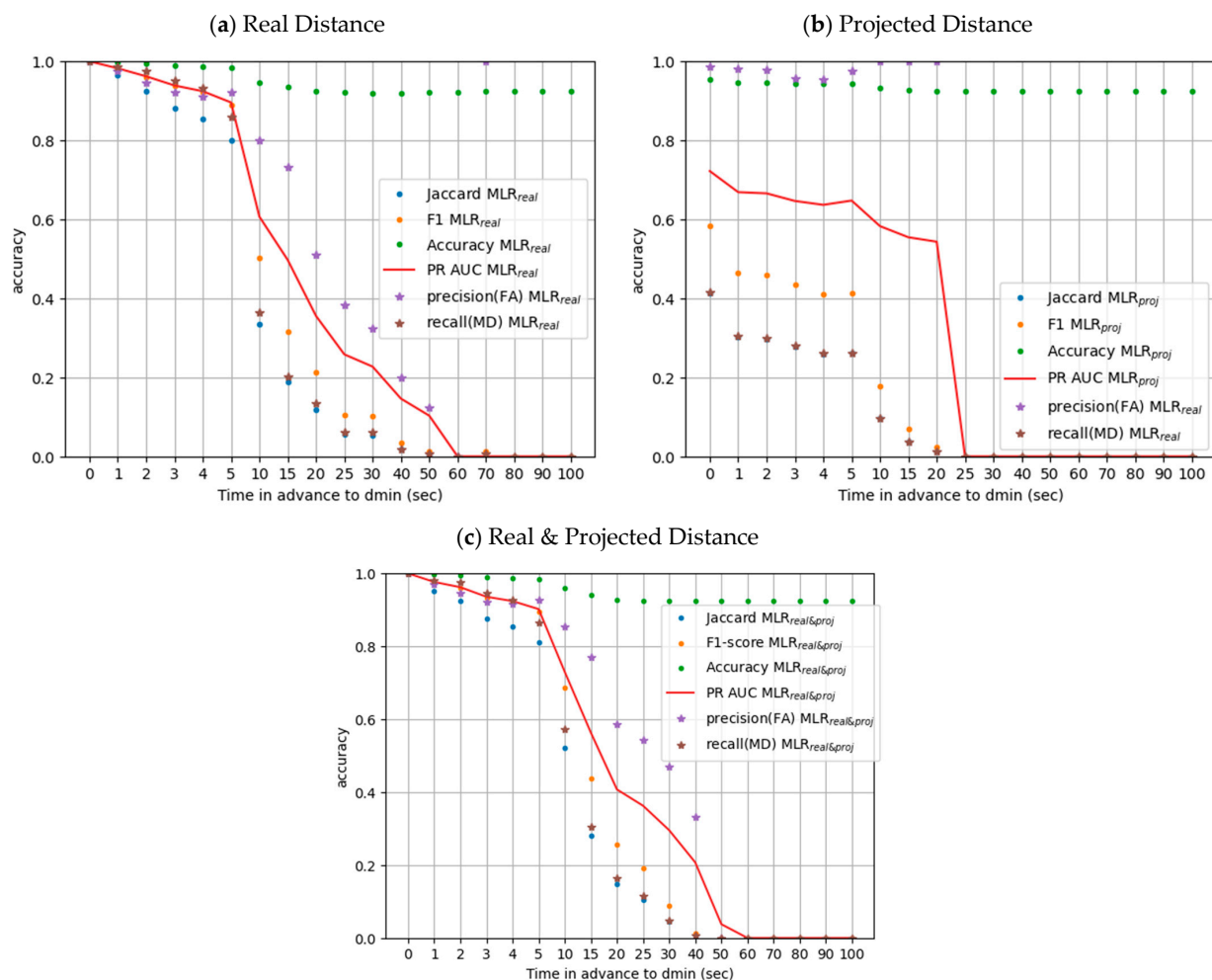


Figure A1. MLR Model Evaluation Metrics Results.

These results show that the classification based on MLR Regression only performs reasonably well up to 5 s in advance before the closest point of approach and only for the models including the real distance; the one based only on projected distances presents very poor results. Figure A2 compares the results for the three models.

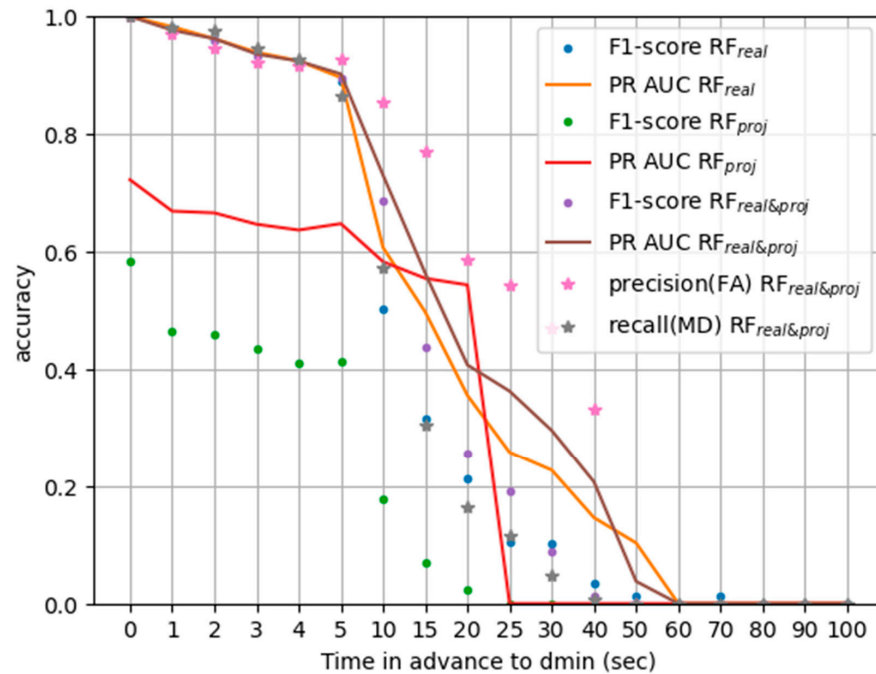


Figure A2. MLR models F1-score and PR AUC comparison.

Appendix A.2. Logistic Regression

The second algorithm implemented is a logistic regression. This is a classification model, as explained in Section 2.1, but based on an underlying regression.

To optimize this model for an imbalanced dataset, it is essential to adjust the weights of the different classes, in this case, 0 and 1. Figure A3 shows the results of the optimization performed to choose the most appropriate class weight for each model.

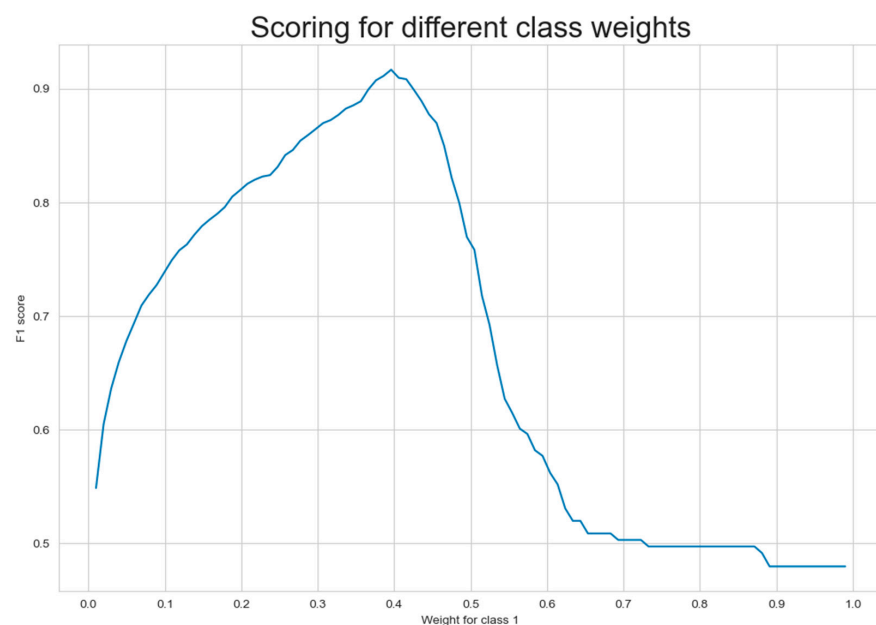


Figure A3. Logistic Regression class weights selection based on F1-score.

The algorithm has been implemented for the three models defined. Figure A4 shows the results for each one.

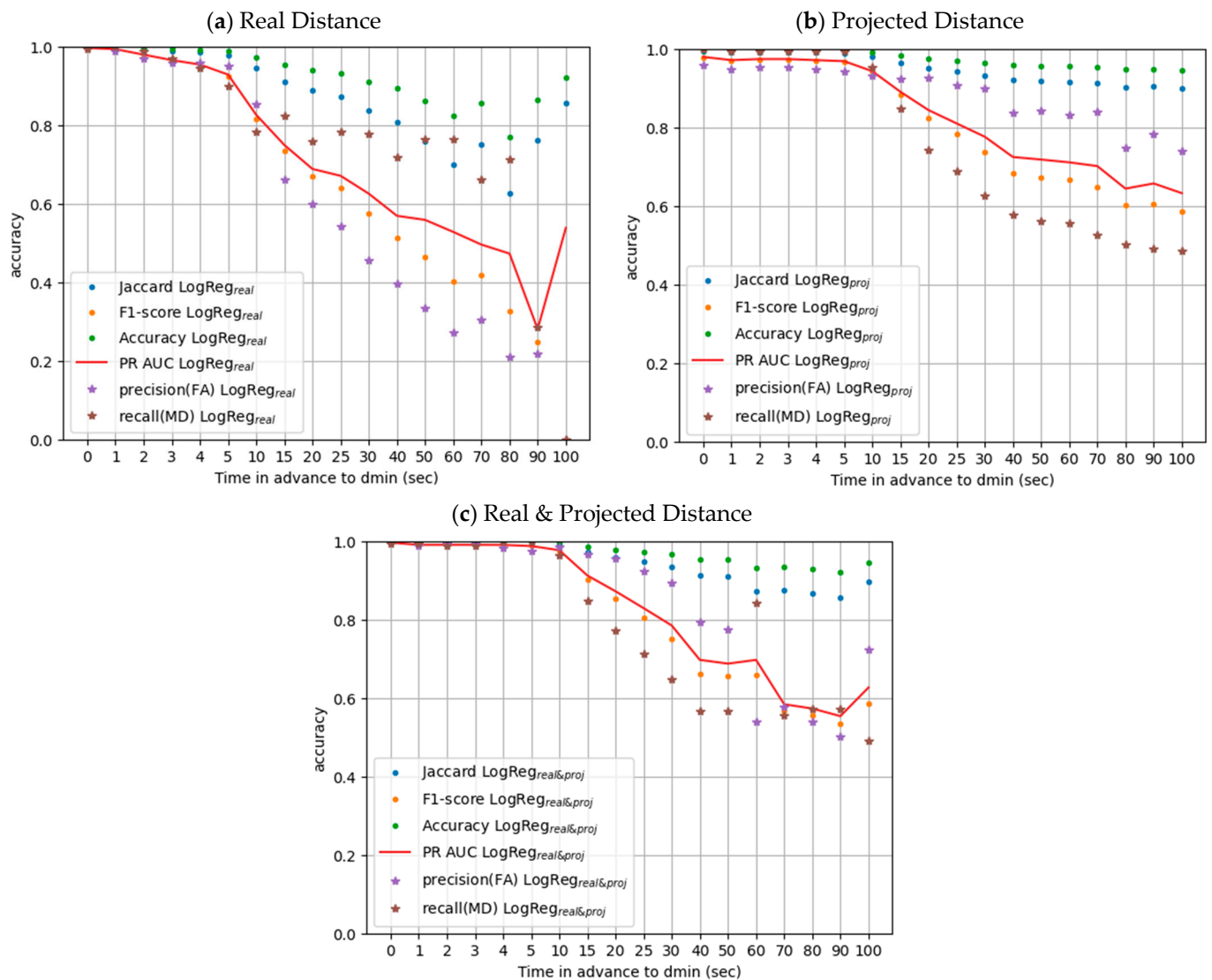


Figure A4. Logistic Regression Models Evaluation Metrics Results.

In this case, the model based on real distances only performs better than the one based on projected distances till 2 s before the minimum distance timespot. The model combining both real and projected distances performs better than both of them till 30 s in advance, although from that moment backwards, the model based on projected distances works better, although presenting poor results in any case. Figure A5 compares the results for the three models.

In the combined model, Precision, Recall and F1-score values are higher than 0.975 up to 5 s in advance before the instant of the closest point of approach; these values represent Missed Detections and False Alarms tending to zero. The Precision, and F1-score remain above this threshold up to 10 s and reasonably high up to 30 s, although in our case, as explained, Recall (missed detections) is more important than Precision (false alarms); however, this evolution could perhaps be reversed to prioritize Recall modifying slightly the hyperparameters of the model.

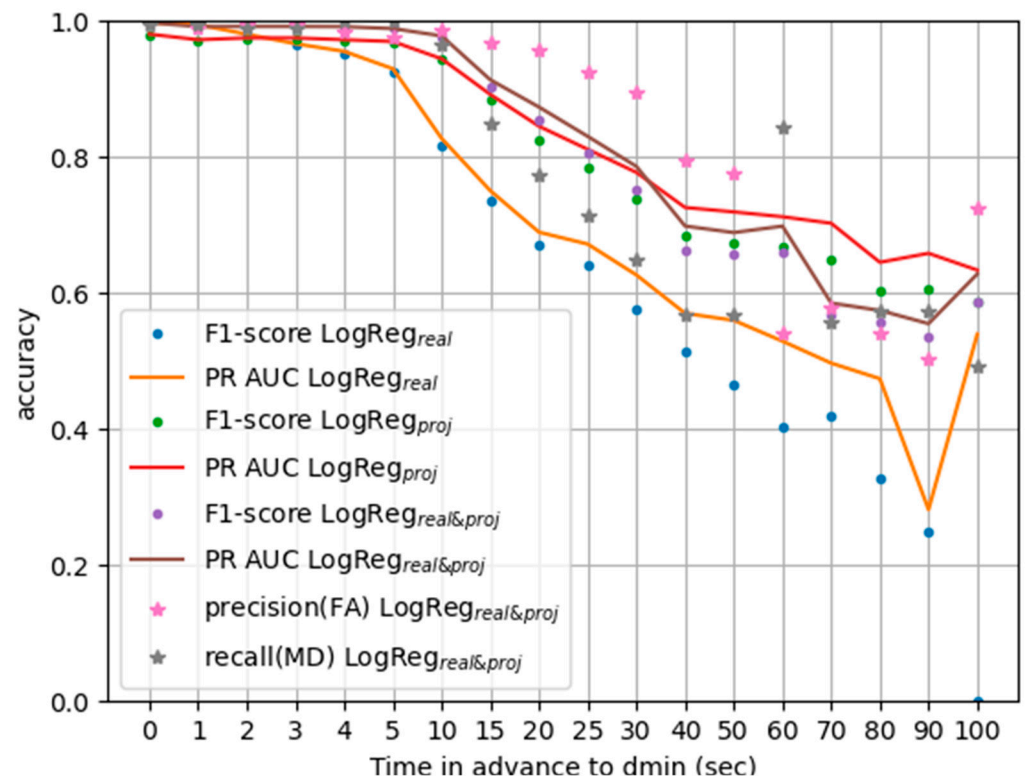


Figure A5. Logistic Regression models F1-score and PR AUC comparison.

Appendix A.3. K-Nearest Neighbours (KNN)

The third algorithm implemented is a KNN classification. The most relevant parameter in this model is the number of neighbors (K) to be considered, so each model has been optimized to maximize the F1-score. Figure A6 shows the results of the optimization performed in one of the samples and Figure A7 the optimum K values for each model and time ahead; it can be observed that the K values are generally odd numbers.

The algorithm has been implemented for the three models defined. Figure A8 shows the results for each one.

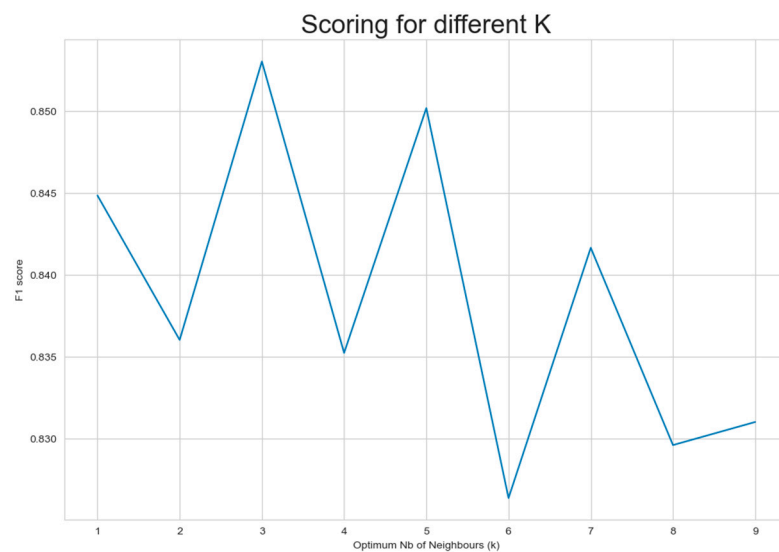


Figure A6. KNN number of neighbors selection based on F1-score.

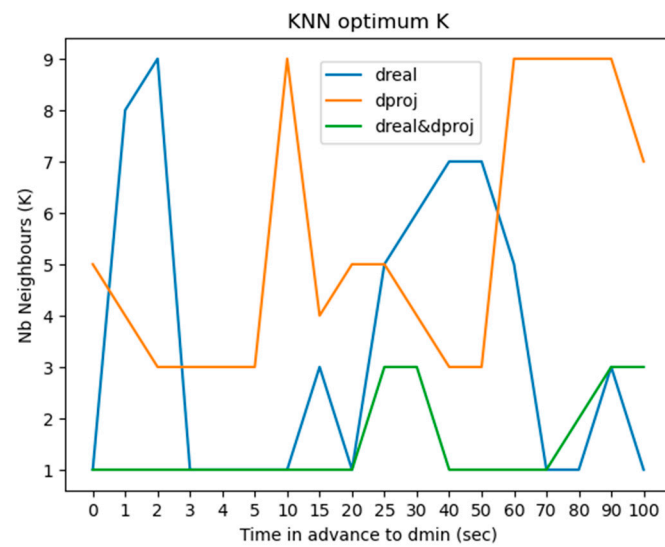


Figure A7. KNN number of neighbors for each model and time.

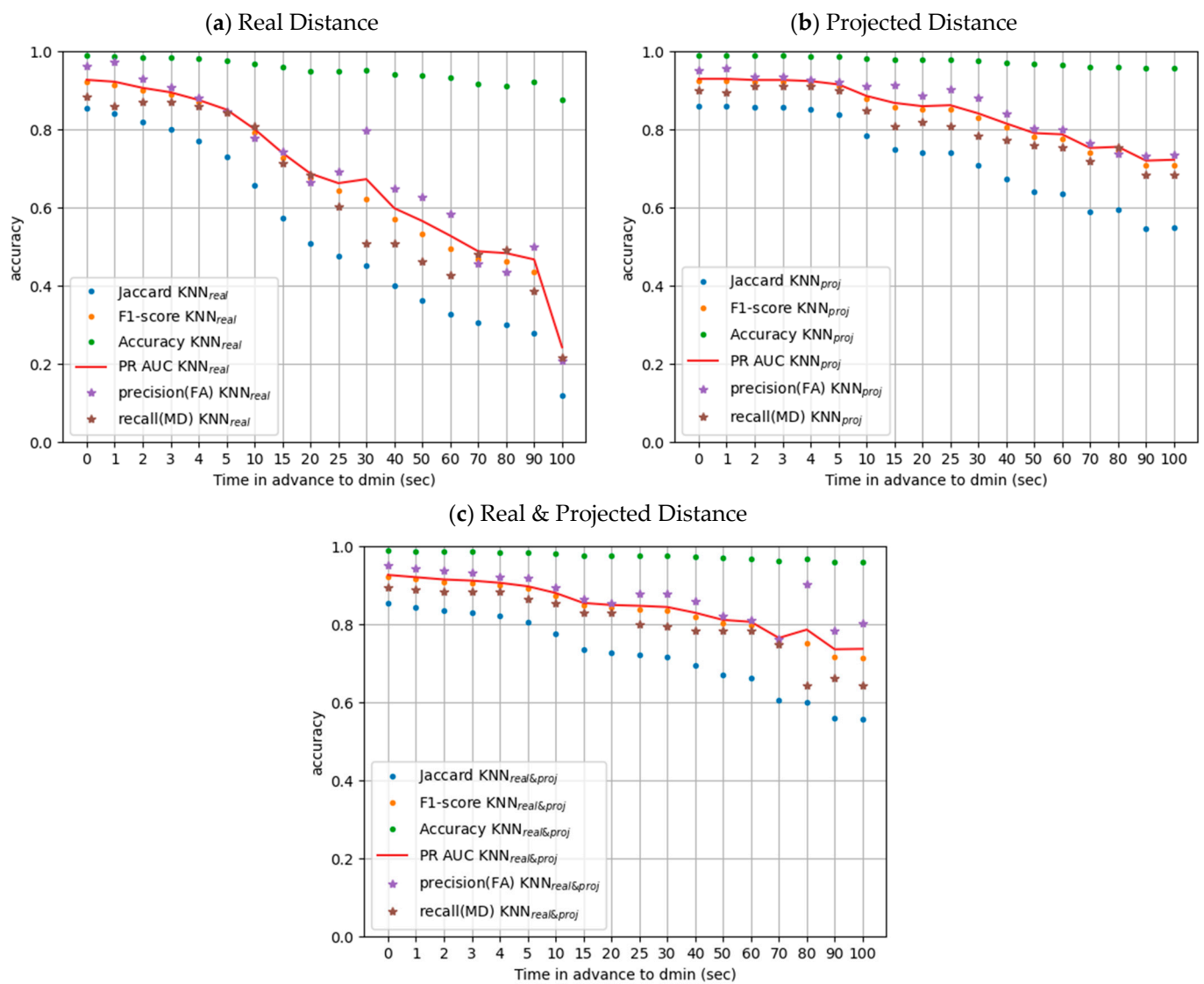


Figure A8. KNN Models Evaluation Metrics Results.

In this case, the model based on projected distances always performs better than the one based on real distances and quite similar to the combined one. These two models do not degrade so sharply with time as in the previous algorithms, performing better than the Logistic Regression from 25 s in advance to the closest point of approach. Figure A9 compares the results for the three models.

However, the model is not sufficiently accurate even for short times from the minimum distance timespot, not being able to detect at least 10% of the Near-Collisions, which increases to 30% as the time to the closest point of approach raises.

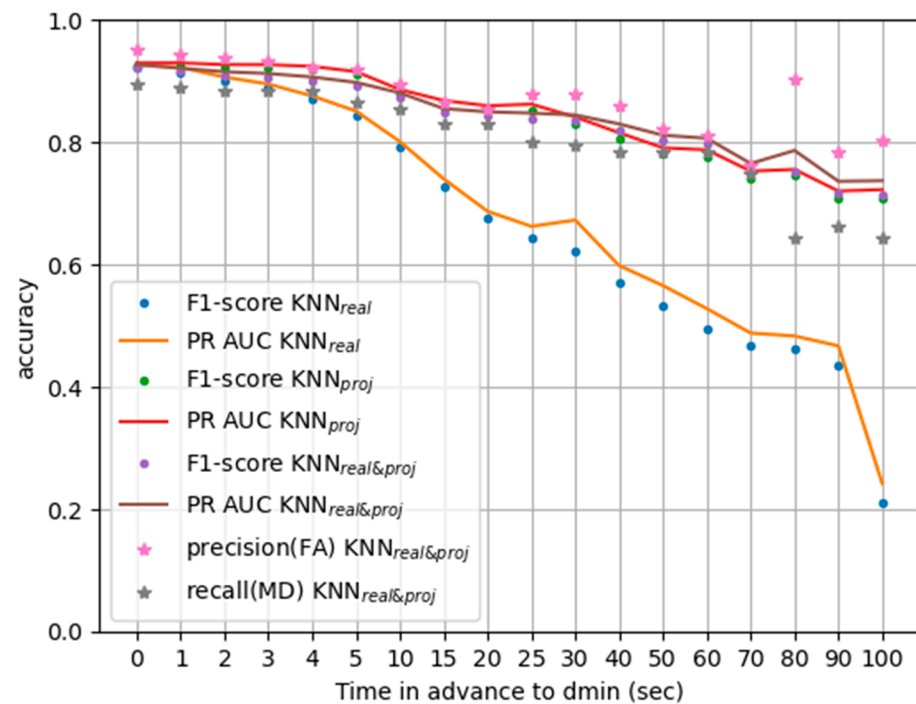


Figure A9. KNN models F1-score and PR AUC comparison.

Appendix A.4. Support Vector Machine (SVM)

The fourth algorithm implemented is a SVM classification. This is a complex model, which includes a number of hyperparameters which, depending on the values taken, determine the performance of the model; therefore, is essential to optimize them in order to provide the most accurate model possible.

To do so, we have applied the GridSearchCV tool provided by the scikit-learn library; this tool develops a search of the optimum tuning of the hyperparameters through an exhaustive evaluation of all parameter combinations.

Table A1 shows the hyperparameters considered in the optimization and the ranges applied. The optimum values identified for tuning depend on the time in advance to the closest point of approach and the type of distances considered.

Table A1. SVM Hyperparameters' search.

Parameter	Meaning	Range Considered in the Optimization
C	Regularization parameter	0.1, 1, 10, 100, 1000
gamma	Kernel coefficient	1, 0.1, 0.01, 0.001, 0.0001
class weight	Weights for each class	{0: w, 1: 1-w} for w in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
kernel	Math function applied to set the boundaries	linear, poly, rbf

The algorithm has been implemented for the three models defined with the optimum hyperparameters identified in the search. Figure A10 shows the results for each one.

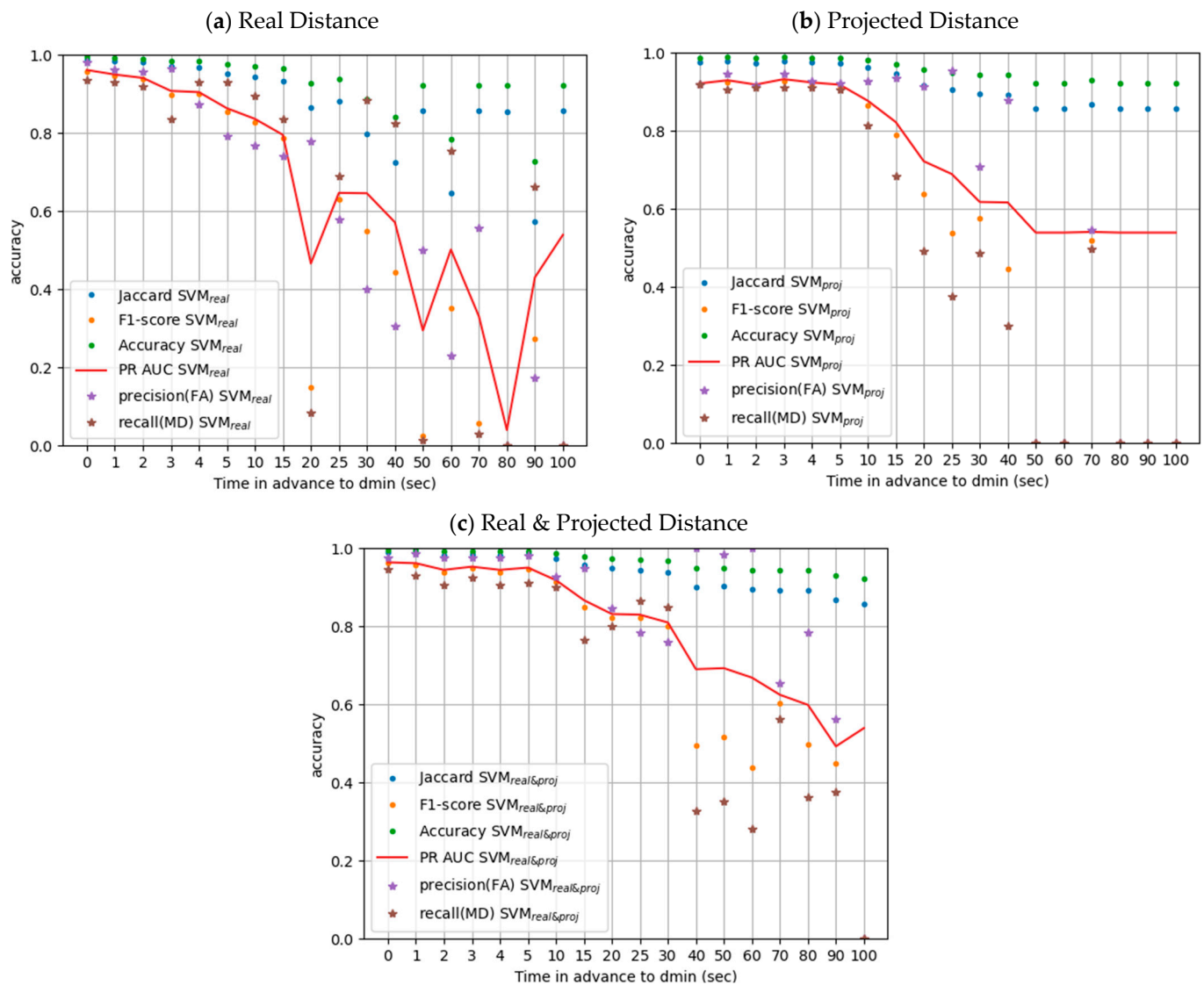


Figure A10. SVM Models Evaluation Metrics Results.

In this case, the model based on real distances only performs better than the one based on projected distances till 2 s before the minimum distance timespot. The model combining both real and projected distances performs better than both of them. Figure A11 compares the results for the three models.

However, the model is not sufficiently accurate even for short times from the minimum distance timespot, not being able to detect more than 5% of the Near-Collisions from 1 s, which increases to more than 50% as the time to the closest point of approach raises.

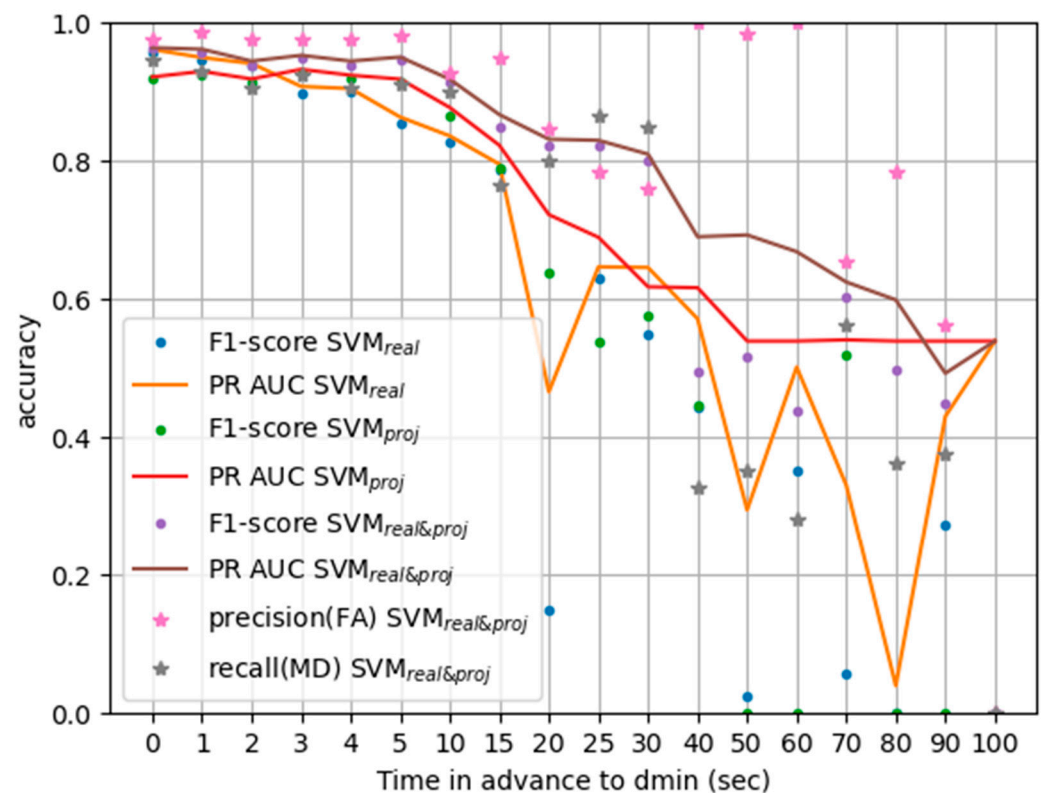


Figure A11. SVM models F1-score and PR AUC comparison.

Appendix A.5. Artificial Neural Network (ANN)

The fifth algorithm implemented is an Artificial Neural Network (ANN), in particular, a Multi-layer Perceptron (MLP) classifier, which is a type of neural network consisting of multiple layers of neurons. This is a supervised learning algorithm that can detect nonlinear relationships in data; naturally, this is a very complex model, in which a lot of hyperparameters must be tuned to improve its accuracy.

To this end, in this model we have also applied the GridSearchCV tool. Table A2 shows the hyperparameters considered in the optimization and the ranges applied. The optimum values identified for tuning depend on the time in advance to the closest point of approach and the type of distances considered.

Table A2. ANN Hyperparameters' search.

Parameter	Meaning	Range Considered in the Optimization
hidden_layer_sizes	Layers and neurons per layer	(10, 30, 10), (20,), (50, 100,)
activation	Activation function for the hidden layer	identity, logistic, tanh, relu
solver	Math function for weight optimization	lbfgs, sgd, adam
alpha	Strength of the regularization	0.0001, 0.05
learning_rate	Learning rate schedule for weight updates	constant, adaptive

The algorithm has been implemented for the three models defined with the optimum hyperparameters identified in the search. Figure A12 shows the results for each one.

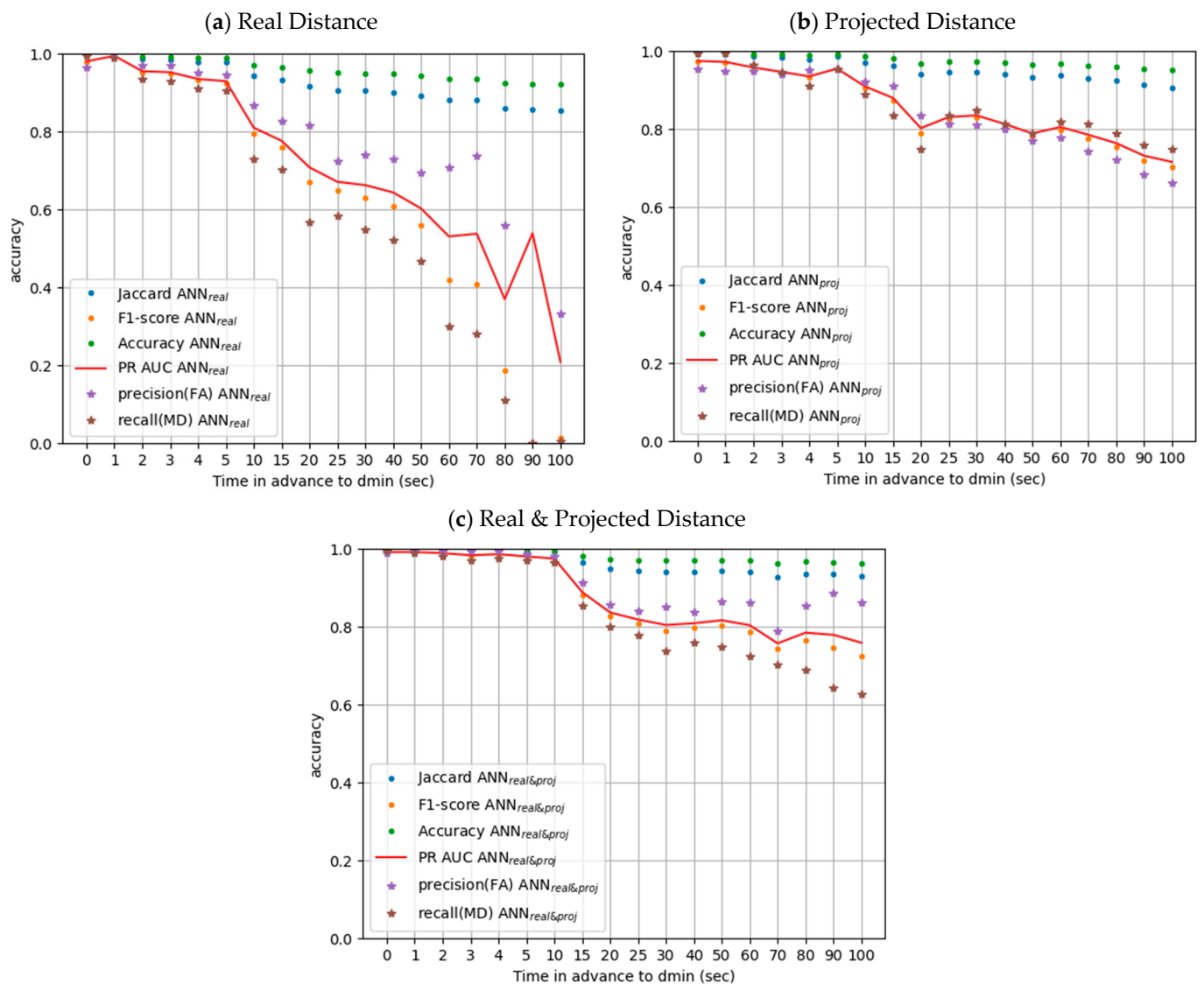


Figure A12. ANN Models Evaluation Metrics Results.

In this case, the model based on real distances only performs better than the one based on projected distances till 2 s before the minimum distance timespot. The model combining both real and projected distances performs better than both of them till 20 s in advance, and from that moment backwards, similarly to the model based on projected distances. Figure A13 compares the results for the three models.

In the combined model, Precision, Recall and F1-score values are higher than 0.975 up to 10 s in advance before the instant of the closest point of approach, worsening slowly with time backwards; these values represent Missed Detections and False Alarms tending to zero in the last moments prior to the near-collision and very small a few seconds before. From 15 s, the F1-score and Recall, present a sharp degradation to values around 0.8 and then, the degradation becomes slower, keeping the F1-score above 0.7 up to 100 s. Again in this model, Recall performs worse than Precision, which is undesirable, but this evolution could perhaps be reversed to prioritize Recall modifying slightly the hyperparameters of the model.

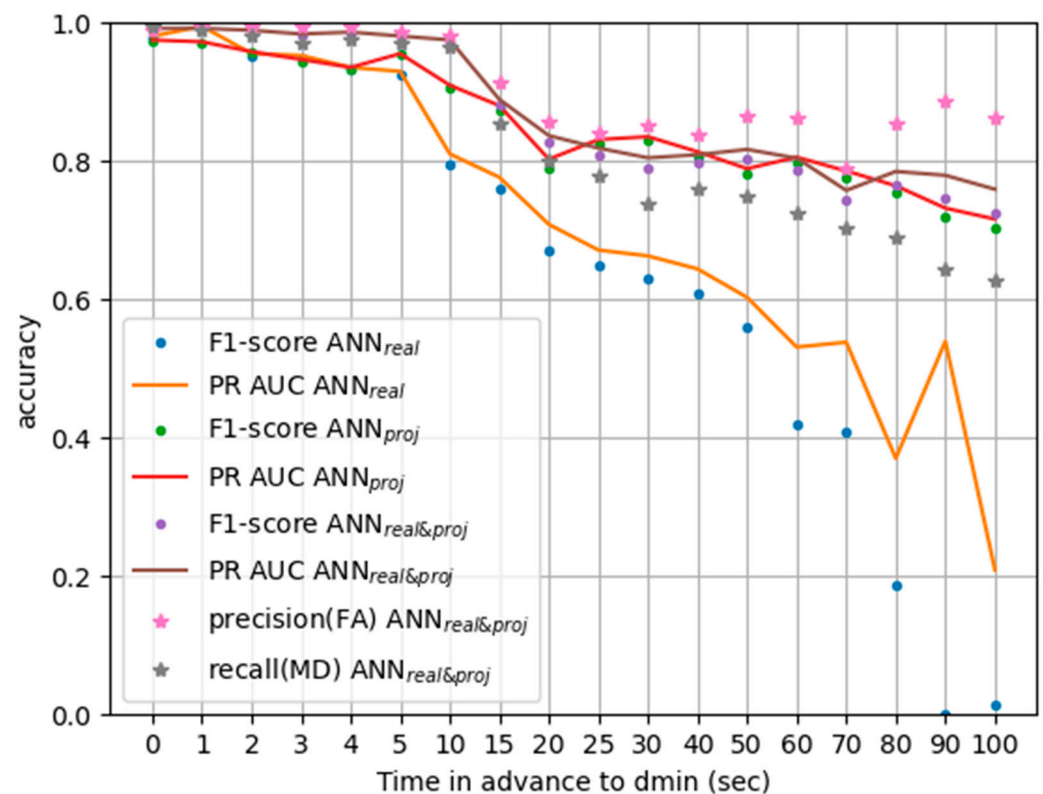


Figure A13. ANN models F1-score and PR AUC comparison.

References

1. The Economist: Report—Civilian Drones. Available online: <https://www.economist.com/technology-quarterly/2017/06/08/drone-technology-has-made-huge-strides> (accessed on 21 July 2024).
2. Kopardekar, P.; Rios, J.; Prevot, T.; Johnson, M.; Jung, J.; Robinson, J.E. Unmanned Aircraft System Traffic Management (UTM) Concept of Operations. In Proceedings of the 16th AIAA Aviation Technology, Integration, and Operations Conference, Washington, DC, USA, 13–17 June 2016. Available online: <https://ntrs.nasa.gov/api/citations/20190000370/downloads/20190000370.pdf> (accessed on 21 July 2024).
3. SESAR Joint Undertaking. European Drones Outlook Study: Unlocking the Value for Europe. 2016. Available online: https://www.sesarju.eu/sites/default/files/documents/reports/European_Drones_Outlook_Study_2016.pdf (accessed on 21 July 2024).
4. Goldman Sachs: Drones. Reporting for Work. Available online: <https://www.washingtonpost.com/sf/brand-connect/goldman-sachs/drones-reporting-for-work/> (accessed on 18 December 2024).
5. Commercial Drone Market Size, Share, and Trends 2024 to 2033. Available online: <https://www.precedenceresearch.com/commercial-drone-market> (accessed on 21 July 2024).
6. Aurambout, J.P.; Gkoumas, K.; Ciuffo, B. Last mile delivery by drones: An estimation of viable market potential and access to citizens across European cities. *Eur. Transp. Res. Rev.* **2019**, *11*, 30. [CrossRef]
7. Groceries Delivered in As Little As 30 Minutes. Available online: <https://wing.com/walmart/> (accessed on 20 July 2024).
8. Irish Drone Delivery Startup Expanding to U.S. and Mainland Europe. Available online: <https://www.forbes.com/sites/johnkoetsier/2023/03/20/irish-drone-delivery-startup-expanding-to-us-and-mainland-europe/> (accessed on 20 July 2024).
9. Amazon Drones Can Now Fly Farther and Deliver to More Customers Following FAA Approval. Available online: <https://www.aboutamazon.com/news/transportation/amazon-drone-prime-air-expanded-delivery-faa-approval> (accessed on 20 July 2024).
10. Sah, B.; Gupta, R.; Bani-Hani, D. Analysis of barriers to implement drone logistics. *Int. J. Logist. Res. Appl.* **2020**, *24*, 531–550. [CrossRef]
11. Reich, C.; Goyal, R.; Cohen, A.; Serrao, J.; Kimmel, S.; Fernando, C.; Shaheen, S. *Urban Air Mobility Market Study*; National Aeronautics and Space Administration: Washington, DC, USA, 2018; pp. 1–163. Available online: <https://ntrs.nasa.gov/citations/20190001472> (accessed on 22 July 2024).
12. Grandl, G.; Ostgathe, M.; Cachay, J.; Doppler, S.; Salib, J.; Ross, H. *The Future of Vertical Mobility: Sizing the Market for Passenger, Inspection, and Goods Services Until 2035*; Porsche Consulting: Stuttgart, Germany, 2018; pp. 1–36. Available online: <https://www.studocu.com/ca-es/document/universitat-politecnica-de-catalunya/infraestructura-y-operacion-de-telecomunicacions/future-of-vertical-mobility/7859863> (accessed on 22 July 2024).

13. Roland Berger. Urban Air Mobility: The Rise of a New Mode of Transportation. 2018. Available online: <https://www.rolandberger.com/en/Insights/Publications/Passenger-drones-ready-for-take-off.html> (accessed on 22 July 2024).
14. Brown, C.; Henig, E.; Anderson, J.; Wilkowski, M.; Taylor, J.; Labib, N.; Taspinar, O.; Reut-Gelbart, M. *Passenger Use Cases in the Advanced Air Mobility Revolution*; KPMG International: Amstelveen, The Netherlands, 2022; pp. 1–23. Available online: <https://kpmg.com/ie/en/home/insights/2022/04/advanced-air-mobility-revolution-aviation.html> (accessed on 22 July 2024).
15. Garrow, L.A.; German, B.J.; Leonard, C.E. Urban air mobility: A comprehensive review and comparative analysis with autonomous and electric ground transportation for informing future research. *Transp. Res. Part C Emerg. Technol.* **2021**, *132*, 103377. [CrossRef]
16. Van Audenhove, F.-J.; Rominger, G.; Korn, A.; Bettati, A.; Steylemans, N.; Zintel, M.; Smith, A.; Haon, S. The Future of Mobility 3.0. Available online: https://www.adlittle.com/sites/default/files/viewpoints/adl_uitp_future_of_mobility_3.0_1.pdf (accessed on 2 July 2024).
17. ICAO. *Annex 2—Rules of the Air*, 10th ed.; ICAO: Montreal, QC, Canada, 2005.
18. Bauranov, A.; Rakas, J. Designing airspace for urban air mobility: A review of concepts and approaches. *Prog. Aerosp. Sci.* **2021**, *125*, 100726. [CrossRef]
19. Barrado, C.; Boyero, M.; Bruculeri, L.; Ferrara, G.; Hatelly, A.; Hullah, P.; Martin-Marrero, D.; Pastor, E.; Rushton, A.P.; Volkert, A. U-Space Concept of Operations: A Key Enabler for Opening Airspace to Emerging Low-Altitude Operations. *Aerospace* **2020**, *7*, 24. [CrossRef]
20. ICAO. Doc 9854-AN/458—Global Air Traffic Management Operational Concept. *International Civil Aviation Organization*. Ed.1. 2005. Available online: [https://www.icao.int/Meetings/anconf12/Document%20Archive/9854_cons_en\[1\].pdf](https://www.icao.int/Meetings/anconf12/Document%20Archive/9854_cons_en[1].pdf) (accessed on 25 July 2024).
21. EU. Commission Implementing Regulation (EU) 2021/664 of 22 April 2021 on a Regulatory Framework for the U-Space. OJ L 139; 23.4.2021. Available online: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32021R0664> (accessed on 25 July 2024).
22. Rios, J. *UAS Traffic Management (UTM) Project Strategic Deconfliction: System Requirements Final Report*; Tech. Rep. NASA Report; NASA: Moffet Field, CA, USA, 2018.
23. ASTM F3548-21; Specification for UAS Traffic Management (UTM) UAS Service Supplier (USS) Interoperability. ASTM International: West Conshohocken, PA, USA, 2022. [CrossRef]
24. Verma, S.; Monheim, S.C.; Moolchandani, K.A.; Pradeep, P.; Cheng, A.W.; Thippavong, D.P.; Dulchinos, V.L.; Arneson, H.; Lauderdale, T.A.; Bosson, C.S.; et al. Lessons Learned: Using UTM Paradigm for Urban Air Mobility Operations. In Proceedings of the AIAA/IEEE 39th Digital Avionics Systems Conference (DASC), San Antonio, TX, USA, 11–15 October 2020; pp. 1–10. [CrossRef]
25. Sacharny, D.; Henderson, T.C.; Cline, M.; Russon, B.; Guo, E. FAA-NASA vs. Lane-Based Strategic Deconfliction. In Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), Virtual, 14–16 September 2020; pp. 13–18.
26. Evans, A.; Anand, A.; Campbell, S.; Anagnostou, A.; Tiberia, S. Simulating Safety and Efficiency Impacts of Airspace Constraints in U-Space Airspace. In Proceedings of the IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC), Barcelona, Spain, 1–5 October 2023; pp. 1–10.
27. Watkins, L.; Sarfaraz, N.; Zanolongo, S.; Silbermann, J.; Young, T.; Sleight, R. An Investigative Study Into An Autonomous UAS Traffic Management System For Congested Airspace Safety. In Proceedings of the IEEE International Workshop on Communication, Computing, and Networking in Cyber-Physical Systems (CCNCPS), Montreal, QC, Canada, 14–18 June 2021.
28. Conte, C.; de Alteriis, G.; Schiano Lo Moriello, R.; Accardo, D.; Rufino, G. Drone Trajectory Segmentation for Real-Time and Adaptive Time-Of-Flight Prediction. *Drones* **2021**, *5*, 62. [CrossRef]
29. Hashemi, S.M.; Botez, R.M.; Ghazi, G. Robust Trajectory Prediction Using Random Forest Methodology Application to UAS-S4 Ehécatl. *Aerospace* **2024**, *11*, 49. [CrossRef]
30. EUROCAE. *Minimum Operational Performance Standards for Traffic Alert and Collision Avoidance System II (TCAS II)*; EUROCAE: Malakoff, France, 2008; Volume I, ED-143.
31. DACUS Consortium. *D4.2—DACUS: Validation Test Results, KPI, and Suitability Metrics & Report*; SESAR Joint Undertaking: Brussels, Belgium, 2022.
32. BURDI Consortium. *D3.1—BURDI: Initial CONOPS*; SESAR Joint Undertaking: Brussels, Belgium, 2023.
33. Learn Classification Algorithms Using Python and Scikit-Learn. Available online: <https://developer.ibm.com/learningpaths/learning-path-machine-learning-for-developers/learn-classification-algorithms/> (accessed on 10 August 2024).
34. Overview of Classification Methods in Python with Scikit-Learn. Available online: <https://stackabuse.com/overview-of-classification-methods-in-python-with-scikit-learn/> (accessed on 10 August 2024).
35. Kozlovskaja, N.; Zaytsev, A. Deep Ensembles for Imbalanced Classification. In Proceedings of the IEEE 16th International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 908–913. [CrossRef]
36. Analyzing Machine Learning Model Performance. Available online: <https://cloud.ibm.com/docs/watson-knowledge-studio?topic=watson-knowledge-studio-evaluate-ml> (accessed on 12 August 2024).
37. Common Classification Model Evaluation Metrics. Available online: <https://towardsdatascience.com/common-classification-model-evaluation-metrics-2ba0a7a7436e> (accessed on 12 August 2024).

38. Gaudreault, J.G.; Branco, P.; Gama, J. An Analysis of Performance Metrics for Imbalanced Classification. In *Discovery Science. DS 2021*; Lecture Notes in Computer Science; Soares, C., Torgo, L., Eds.; Springer: Cham, Switzerland, 2021; Volume 12986. [[CrossRef](#)]
39. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res. (JMLR)* **2011**, *12*, 2825–2830.
40. Peinecke, N.; Limmer, L.; Volkert, A. Application of “Well Clear” to Small Drones. In Proceedings of the IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), London, UK, 23–27 September 2018; pp. 1–9. [[CrossRef](#)]
41. Weinert, A.; Campbell, S.; Vela, A.; Schuldt, D.; Kurucar, J. Well-clear recommendation for small unmanned aircraft systems based on unmitigated collision risk. *J. Air Transp.* **2018**, *26*, 113–122. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.