Air Traffic Flow Identification and Recognition in Terminal Airspace through Machine Learning Approaches

Wenxin Zhang*, Alexia P. Payan†, and Dimitri N. Mavris‡
Aerospace Systems Design Laboratory, School of Aerospace Engineering,
Georgia Institute of Technology, Atlanta, GA, 30332-0150, USA

In modern aviation, a significant amount of data is generated during routine operations and collected using technologies like Automatic Dependent Surveillance-Broadcast (ADS-B). The abundance of such data presents great potential for utilizing emerging data analysis techniques like machine learning to enhance the future of aviation. This paper presents a methodology that leverages clustering and classification models for offline identification and online recognition of air traffic flows. This research utilizes real trajectories in the terminal area of Zurich Airport to train and assess various machine learning models. To prepare the raw trajectory data for analysis, we apply a preprocessing step to clean and resample the data. Clustering is performed using the Ordering Points to Identify the Clustering Structure (OPTICS) algorithm, and its performance is compared to Density-Based Spatial Clustering of Applications with Noise (DBSCAN). For classification of the data, we employ two ensemble methods, Random Forest and Extreme Gradient Boosting (XGBoost), and compare their outcomes with those of Long Short-term Memory (LSTM). Our results demonstrate the superior reliability of OPTICS compared to the baseline method for clustering, and the ensemble models perform as effectively as the deep learning model, but with shorter training times due to their relative simplicity. The proposed methodology enhances the understanding of air traffic flows at specific airports and facilitates subsequent trajectory-centric tasks such as anomaly detection, trajectory prediction, and conflict detection, ultimately contributing to the improvement of safety in the terminal airspace.

I. Introduction

In the terminal airspace, Standard Instrument Departures (SID) and Standard Instrument Arrivals (STAR) provide a basic structure of air traffic flows for aircraft operating under Instrument Flight Rules (IFR). However, due to the dynamic nature of operational conditions, real flights often deviate from these standards [1, 2]. Additionally, Air Traffic Controllers (ATCOs) often vector incoming aircraft to adjust their routes for safety or efficiency [3]. Despite these variations, patterns in air traffic flows can be observed in daily terminal airspace operations. Thus, studying air traffic flows has become a significant area of interest for the aviation industry as it supports various applications, including load balancing, anomaly detection, environmental impact assessment, and conflict detection [1, 4, 5]. Recent years have seen a growing concern for Air Traffic Flow Management (ATFM) due to increased traffic volumes and limited resources leading to congestion and safety issues in various regions worldwide [6].

The expansion of historical aircraft trajectory data has sparked significant interest among researchers investigating air traffic flows, particularly in offline trajectory clustering - a field within machine learning and artificial intelligence. Clustering, as an unsupervised learning technique, utilizes extensive historical data and clustering algorithms to form groups or clusters where samples within the same cluster exhibit higher similarity compared to those outside [4, 7]. This approach enables the grouping of flights with similar spatial and temporal characteristics, leading to identifying patterns in air traffic flows.

In addition to offline clustering, real-time recognition of traffic flows is of great importance. This recognition enhances the situational awareness of pilots and controllers, enabling them to evaluate and respond promptly to potentially hazardous conditions [8]. Aviation organizations are also actively working to extend offline clustering techniques to online applications [9]. The primary difference between offline and online air traffic flow identification

*Ph.D. Candidate, Aerospace Systems Design Laboratory, School of Aerospace Engineering, 270 Ferst Drive, Atlanta, GA, 30332.
†Research Engineer II, Aerospace Systems Design Laboratory, School of Aerospace Engineering, 270 Ferst Drive, Atlanta, GA, 30332-0150
‡S.P. Langley Distinguished Regents Professor, Boeing Regents Professor of Advanced Aerospace Systems Analysis, School of Aerospace Engineering, Director Aerospace Systems Design Laboratory, 270 Ferst Drive, Atlanta, GA, 30332-0150, AIAA Associate Fellow.

lies in the data provided to the model. Offline learning utilizes complete trajectory data, while online learning relies on partial trajectories for predicting the air traffic flow of a flight. The results obtained can further support tasks such as anomaly detection and conflict detection, thus improving overall aviation operations.

The structure of this paper is given as follows: In Section II, an overview of the relevant research regarding the study and analysis of air traffic flows is provided. In Section III, the methodology developed in this study is outlined. In Section IV, the dataset used in this research is introduced, along with the preprocessing workflow applied to it. In Section V, the implementation details of the clustering and classification methods utilized are presented. In Section VI, we compare the machine learning models developed and present the corresponding main results. In Section VII, the major findings of this research are summarized.

II. Literature Review

Trajectory clustering relies on the utilization of clustering algorithms and distance metrics to evaluate the similarity between trajectories [3, 10]. In their study of traffic patterns in the Bay Area, Gariel et al. [11] employed K-means and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithms, with trajectory features extracted through Principle Component Analysis (PCA). K-means assigns points to cluster centers iteratively and adjusts the centers to their means, whereas DBSCAN separates dense clusters from sparse noise using a specified search distance and does not necessitate a predefined number of clusters [12]. Basora & Mailhot [7] proposed an automated framework that utilized Historical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN), a modified version of DBSCAN, that requires a reduced number of input parameters. The authors explored distance metrics such as Euclidean Distance (ED) and Symmetrized Segment-Path Distance (SSPD). ED is a straightforward mathematical distance that can be applied easily, but it can only compare trajectories of equal lengths. On the contrary, SSPD is a shape-based distance that remains unaffected by time (and thus trajectory length), making it robust against incidental variations between trajectories, albeit requiring more computational resources. Other approaches include Longest Common Sub-Sequence (LCSS) with spectral clustering [8], weighted ED combined with HDBSCAN [13], and clustering based on Dynamic Time Warping (DTW) [3]. It is noteworthy that the extended version of DBSCAN called Ordering Points to Identify the Clustering Structure (OPTICS) has not been extensively explored in aviation trajectory clustering research.

There is a limited body of literature available that explores the use of supervised learning techniques for real-time recognition of air traffic flows. Bosson and Nikoleris [14] applied various supervised techniques to classify and predict on which runway flights would land. They experimented with classifiers such as Logistic Regression, Support Vector Machines (SVM), Bayes classifiers, K-Nearest Neighbors (KNN), Decision Trees, as well as ensemble methods like Random Forest (RF) and AdaBoost. Additionally, they employed deep learning algorithms, specifically Multilayer Perceptron (MLP) and Convolutional Neural Networks (CNN) and compared the performance of these prediction models in terms of accuracy and runtime. They concluded that simpler classifiers such as DT and KNN required less training time compared to more sophisticated models like RF and MLP, but at the expense of performance. Hong and Lee [3] proposed a probabilistic method for predicting the estimated time of arrival of incoming flights. Morris and Trevedi [8] presented a framework for classifying GPS-based trajectories and detecting unusual actions in real-time. They employed a sliding window approach during the online prediction phase, considering only the recent history of the trajectory instead of the entire past trajectory. Madar et al. [2] utilized RF to perform classification and recognition of traffic flows of arrival flights and achieved accurate results.

To summarize, offline trajectory clustering requires the selection of a distance metric to assess trajectory similarity and a suitable algorithm to group trajectories accordingly. The Euclidean Distance (ED) metric is frequently employed in trajectory clustering due to its simplicity and computational efficiency. While some studies have utilized partitional clustering algorithms like K-means, the main challenge lies in the difficulty of determining the total cluster number in advance. On the other hand, density-based methods present alternatives that circumvent this challenge. Moreover, density-based clustering effectively filters out noise, such as trajectories that deviate from standard operating procedures, thereby improving the quality of clustering results since noise is commonly encountered in daily operations. Although a few studies have focused on online trajectory analysis aimed at predicting metrics like estimated time of arrival and landing runway, there is a need to explore real-time recognition of air traffic flows. Indeed, the prediction of a flight traffic flow would yield significant benefits for tasks such as anomaly detection and conflict detection.

III. Methodology

A schematic overview of the proposed methodology, its outcomes, and the interconnections between modules is provided in Fig. 1. The methodology consists of two sequential modules: the clustering module and the classification module, which make use of separate machine learning algorithms. In the clustering module, offline clustering methods are applied to examine historical ADS-B data and identify the distinctive traffic patterns within terminal airspace. Drawing upon the knowledge acquired, the classification module utilizes classification methods to decide the real-time traffic flow to which incoming landing flight trajectories belong, with the capability to also identify trajectories that deviate from regular traffic flows.

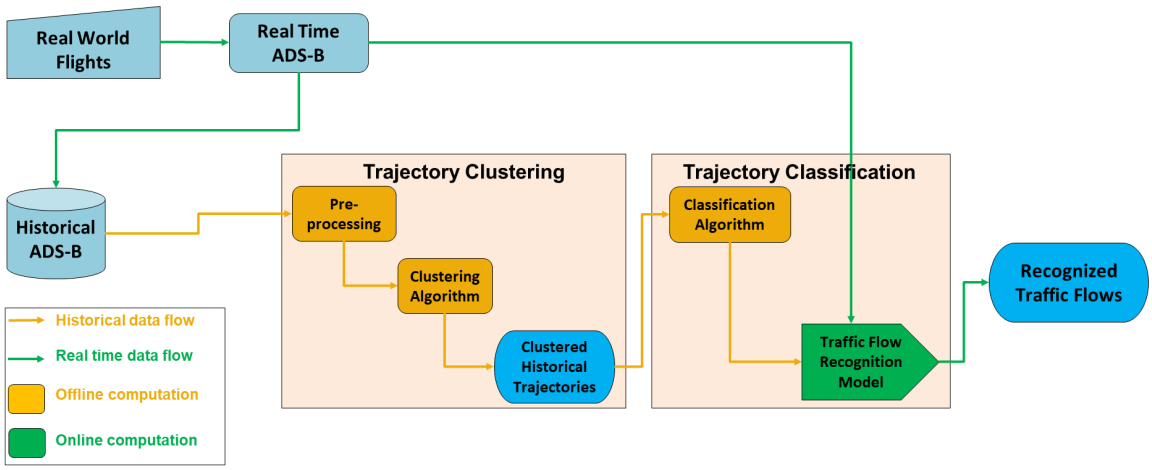


Fig. 1 Methodology Overview.

Trajectory data in aviation contains random components and noise, which can be challenging to filter out through partitioning or hierarchical approaches. On the other hand, density-based clustering methods are effective in noise elimination. The most popular density-based clustering algorithm in the aviation literature is DBSCAN [2, 7, 11], which separates dense clusters from noise based on a distance threshold and a minimum number of samples. However, DBSCAN has limitations in handling clusters with varying density levels and is sensitive to input parameters. OPTICS [15] overcomes these limitations by creating an ordering of data points based on core and reachability distances, allowing for the identification of clusters with different density levels and reducing sensitivity to parameters. In addition, OPTICS provides a reachability plot that visualizes the cluster structure and aids in understanding the clustering process. Hence, OPTICS is proposed for trajectory clustering to identify air traffic flows in the terminal airspace. However, DBSCAN is still used as a baseline model and its results are compared to those of OPTICS.

Real-time air traffic flow recognition can be approached as a multi-class classification problem, where a trajectory segment serves as input and the corresponding air traffic flow is the target. The aviation literature has explored various supervised learning techniques for classification, including efficient yet less accurate models such as LR, SVM, Naive Bayes, and DT. Deep learning models such as MLP and CNN offer higher accuracy but require more computational resources. Ensemble models like RF and XGBoost combine multiple base models to provide stable and accurate predictions. RF utilizes many decision trees and the bagging strategy, while XGBoost employs the boosting strategy to integrate results from the base models [16, 17]. These ensemble models have the potential to achieve accurate predictions with faster training times compared to deep learning models. Hence, RF and XGBoost are proposed as classifiers for air traffic flow recognition, prioritizing accuracy and efficiency. Additionally, a Long Short-term Memory (LSTM) model [18] will also be developed for comparison with the ensemble models. It is important to mention that the labels for the classification training data are directly derived from the results of the Trajectory Clustering module, as shown in Fig. 1.

IV. Dataset

A. Description

Zurich Airport (ZRH/LSZH) is a bustling commercial airport situated in Europe, serving a substantial number of passengers, with an impressive count of over 30 million in 2019. It operates with three runways dedicated to both landing and departure activities, contributing to the intricate nature of its traffic patterns. This study focuses on a comprehensive analysis of two complete months, specifically October and November of 2019, as they provide a representative sample of the regular traffic patterns observed within the airspace. In light of their potential impact on airspace safety and efficiency, the data used in this research concentrates specifically on landing flights.

The raw ADS-B data utilized in this research is obtained from the OpenSky Network. The dataset encompasses a total of 19,480 landing flights at LSZH, spanning the period between 10/1/2019 and 11/30/2019. Represented in a tabular format, the dataset comprises a vast collection of 18,177,597 rows of records.

B. Preprocessing

The ADS-B dataset cannot be directly used by machine learning algorithms due to potential missing or erroneous data inherent in the collaboratively collected nature of OpenSky Network. Therefore, a comprehensive data preprocessing phase is essential to refine the raw data and make it suitable for the subsequent tasks. Additionally, the clustering algorithm employed in this study requires trajectories of equal length as input, necessitating the resampling of trajectories before they can be used for clustering purposes. Figure 2 provides an overview of the data pre-processing steps, including data cleaning, augmentation, and manipulation. Through this process, the raw ADS-B dataset is transformed into a refined dataset that is well-suited for the classification task, as well as a resampled dataset specifically tailored for clustering. The preprocessing steps make use of the Traffic library [19], a Python library designed for processing and analyzing aircraft trajectory data.

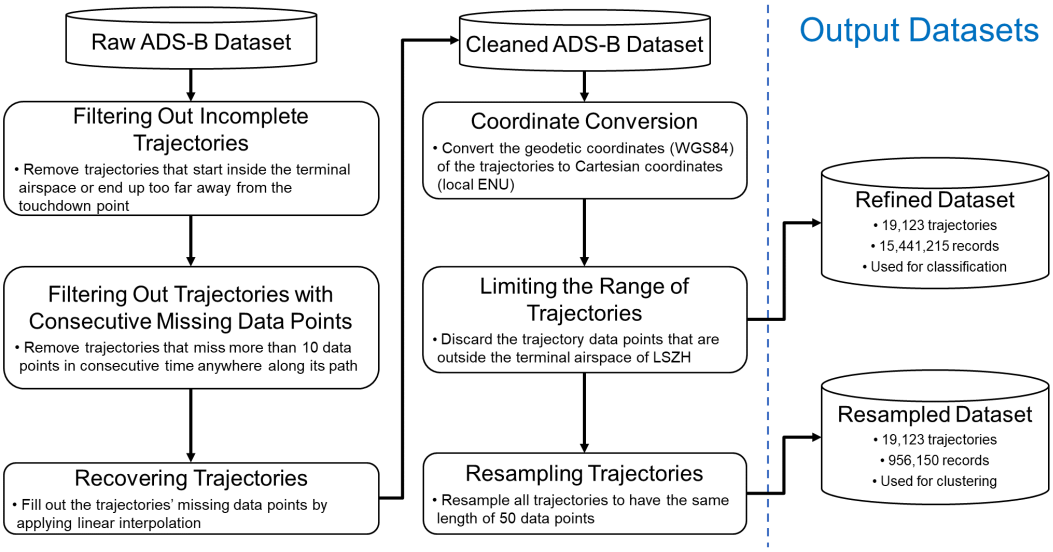


Fig. 2 Data Pre-processing Workflow

The local ENU coordinate system is used to present 800 sample trajectories from the Refined Dataset as shown in Fig. 3. In this visualization, the coordinate system's origin denotes the location of the airport's highest runway.

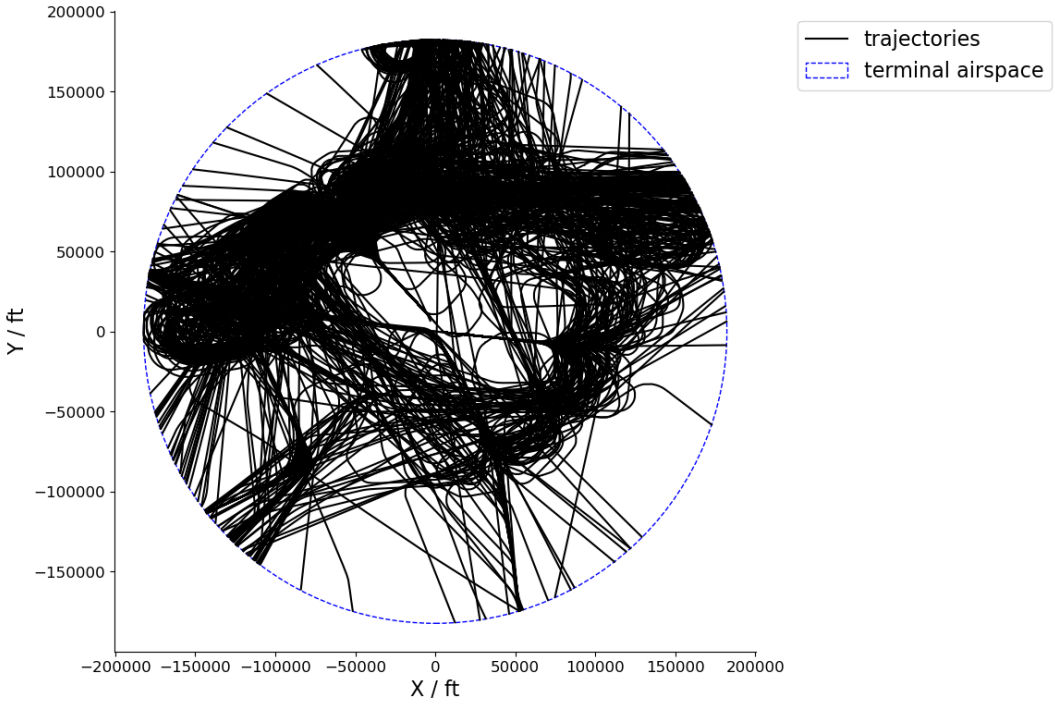


Fig. 3 Sample Trajectories from Refined Dataset (800 samples)

V. Implementation

A. Data Split

Utilizing the Resampled Dataset or the Refined Dataset in its entirety for constructing machine learning models is not ideal due to two issues. Firstly, it would significantly lengthen the training process, and secondly, it would produce a single clustering/classification result set, limiting the ability to conduct comprehensive analyses, especially when comparing outcomes across different machine learning models.

To address these concerns, the trajectory set is divided into multiple subsets based on a weekly basis. This division results in nine distinct subsets of Resampled/Refined Trajectories, denoted as 0, 1, ..., 8, representing specific weeks. The machine learning process is then separately applied to each subset, generating nine distinct result sets. These sets are then utilized to compare and assess the performance of various models in the subsequent analysis, allowing for a more thorough assessment of the models.

B. Clustering

1. Distance Metric

The selected distance metric is the Weighted Euclidean Distance (WED), which is favored for its efficient computation and ability to focus on the most crucial segments of trajectories. The mathematical expression of the WED between two trajectories, denoted as T_a and T_b, is as follows:

$$D_{WED}^{T_a,T_b} = \sqrt{\sum_{i=1}^{n} w_i \sum_{f=1}^{k} (T_a^{i,f} - T_b^{i,f})^2} \tag{1}$$

Where the weight assigned to each point in a trajectory at index i is represented by w_i, allowing for customization according to the user's preference. To emphasize the "middle part" of the trajectory, a beta distribution is employed as the weight scheme. This approach has proven to be robust for weighting trajectories in clustering analyses, as evidenced in [13]. Specifically, the beta distribution parameters are selected as $\alpha = 1.8$ and $\beta = 3$. Figure 4 displays the weight

scheme based on the beta distribution, alongside a uniform scheme that results in the original ED. Notably, the beta distribution emphasizes the points located in the middle of the trajectory.

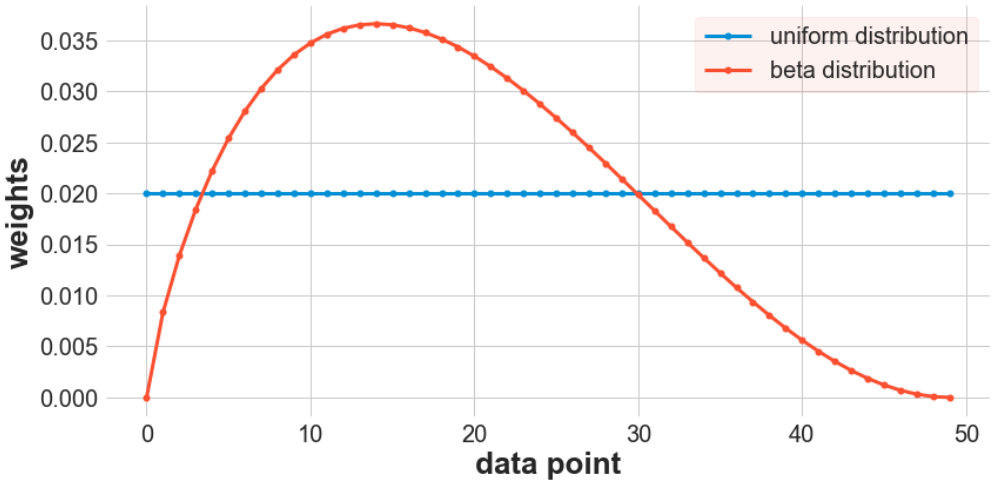


Fig. 4 Weight Scheme Comparison

2. Clustering Algorithm

Two algorithms, namely DBSCAN and OPTICS, are used to perform clustering. WED is used for pre-computing distances between all pairs of input trajectories. Scikit-learn [20], an open-source Python library for machine learning, is utilized to implement the two clustering algorithms.

DBSCAN requires two user-defined parameters, namely ϵ and $MinPts$, whereas OPTICS only requires the $MinPts$ parameter. The selection of these parameters can impact the clustering outcomes, particularly the rate of outliers. Thus, an iterative approach is employed to determine the most suitable parameter settings. Research conducted by Gariel et al. [11] on the daily rate of outlier trajectories at San Francisco International Airport revealed a range of 2% to 16%. Considering the focus of this research is Zurich Airport and the dataset is based on a weekly basis, the upper bound is adjusted to 25% after analyzing the dataset.

The first step consists im executing OPTICS, which requires searching for the optimal $MinPts$ parameter. An iterative search is conducted by exploring a range of $MinPts$ values. The search space is set between 1.5% and 2.5% of the number of trajectories in the associated weekly dataset. If multiple parameter settings meet the outlier rate requirement, the one with the lowest outlier rate is selected. In cases where none of the parameter combinations fulfill the outlier rate requirement, the week is excluded from further analysis.

The second step focuses on determining the best parameters for the DBSCAN algorithm. To ensure comparability with the OPTICS results, the $MinPts$ parameter in DBSCAN is set to the same value as in OPTICS. Consequently, the search becomes one-dimensional, with the epsilon parameter varying between 5% and 15% of the maximum distance between any two trajectories in the weekly dataset. If multiple parameter combinations satisfy the outlier rate requirement, the one closest to the outlier rate obtained from OPTICS is chosen. If none of the parameter combinations meet the outlier rate requirement, the week is excluded from further analysis.

Finally, a comparison is made between the resulting clusters obtained from both algorithms for each week.

C. Classification

1. Duration of Input Trajectory

The duration of the input trajectory segment, known as the "observation time" in this study, can be customized to meet specific requirements. The Refined Dataset comprises trajectories of different lengths, as depicted in the overview provided in Fig. 5. The median length is approximately 750, with half of the trajectories falling within the range of 600 to 900 points, which corresponds to a duration of 10 to 15 minutes considering the one-second intervals of ADS-B data. Based on this information, setting the observation time to a few minutes would be reasonable. To evaluate the impact of

this variable on the model, several observation times were tested, including 60, 120, 180, and 240 seconds. Observation times exceeding 240 seconds are considered excessively long and would result in a model with inadequate predictive power.

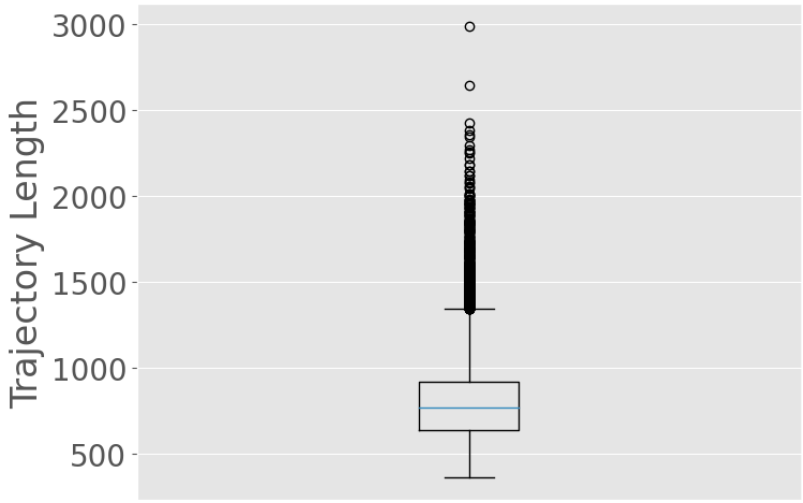


Fig. 5 Trajectory Length Overview

2. Trajectory Classification

For the traffic flow recognition task, three models are developed: two ensemble models (RF and XGBoost) and a deep learning model (LSTM). The ensemble models, RF and XGBoost, are implemented using the Scikit-learn library, while the deep learning model, LSTM, is implemented in PyTorch [21], which is a highly optimized framework for deep learning in Python, widely used in both academic and industrial environments.

Hyperparameter Tuning

Hyperparameters in a machine learning model play a crucial role in determining its performance by controlling the learning process. To ensure a fair comparison, hyperparameter tuning is conducted for each model type to find the optimal set of hyperparameters that maximize performance.

Grid search and random search are commonly used methods for hyperparameter tuning, but they can be computationally expensive and have limitations. Bayesian optimization [22] is a more advanced approach that constructs a probabilistic model and employs Bayesian reasoning to guide the search for promising hyperparameter sets. This method is particularly beneficial for complex models with expensive objective functions. In this study, Optuna [23], a Python-based automatic hyperparameter optimization framework, is utilized, employing a specific implementation of Bayesian optimization known as the Tree-structured Parzen Estimator (TPE) algorithm.

In our implementation, the RF model incorporates three key parameters: *n_estimators*, *max_depth*, and *min_samples_split*. Similarly, the XGBoost model involves three primary parameters: *learning_rate*, *max_depth*, and *subsample*. The search space for each parameter is defined as a range, allowing for exploration within specific intervals. In contrast, the LSTM model encompasses five tunable parameters: *batch_size*, *num_layers*, *hidden_size*, *dropout*, and *learning_rate*. The search space for *batch_size*, *num_layers*, and *hidden_size* is defined as a set of discrete integer numbers, while the search space for *dropout* and *learning_rate* is represented as ranges. The determination of these search spaces relies on heuristic techniques. For a comprehensive overview of the search spaces for the classification models, please refer to Table 1.

Train-test Split and Cross-validation

The classification models are trained on a weekly basis, with input labels derived from weekly clustering. For the ensemble methods, the weekly trajectories are divided into training/testing set, following an 80-20 split convention. The training set is employed to construct the model, while the testing set evaluates model performance. For the LSTM

Table 1 Hyperparamter Search Space

Model Type	Hyperparameter	Min	Max	Data Type	Search Set
RF	n_estimators	20	200	Int	-
	max_depth	2	20	Int	-
	min_samples_split	2	20	Int	-
XGBoost	learning_rate	1e-4	2e-1	Float	-
	max_depth	2	20	Int	-
	subsample	0.4	1	Float	-
LSTM	batch_size	-	-	-	[16, 32, 64, 128]
	num_layers	-	-	-	[1, 2, 4]
	hidden_size	-	-	-	[8, 16, 32, 64, 128]
	dropout	0	0.5	Float	
	learning_rate	1e-4	2e-1	Float	

model, the training set is further divided into train and validate subsets. During training iterations, the LSTM model updates parameters using the train set and evaluates performance using the validate set. After completing iterations, the testing set assesses the trained model's performance.

To address data variability, noise, and overfitting, we utilize cross-validation with stratified shuffle split. This technique randomly divides the input data into training and testing sets while preserving the original label distribution. In the case of the ensemble models, this process repeats for ten times to train and test ten distinct models. The performance of models on the testing set is then averaged to obtain an overall performance metric. However, due to the considerably longer training time of the LSTM model compared to the ensemble models, we reduce the number of iterations from ten to two for efficiency purposes.

VI. Results

A. Clustering

Out of the nine weeks included in the analysis, Week 4 is excluded due to not meeting the outlier rate requirement, whereas the remaining eight weeks satisfy the outlier rate condition, allowing both algorithms to produce clusters with an outlier rate below 25%.

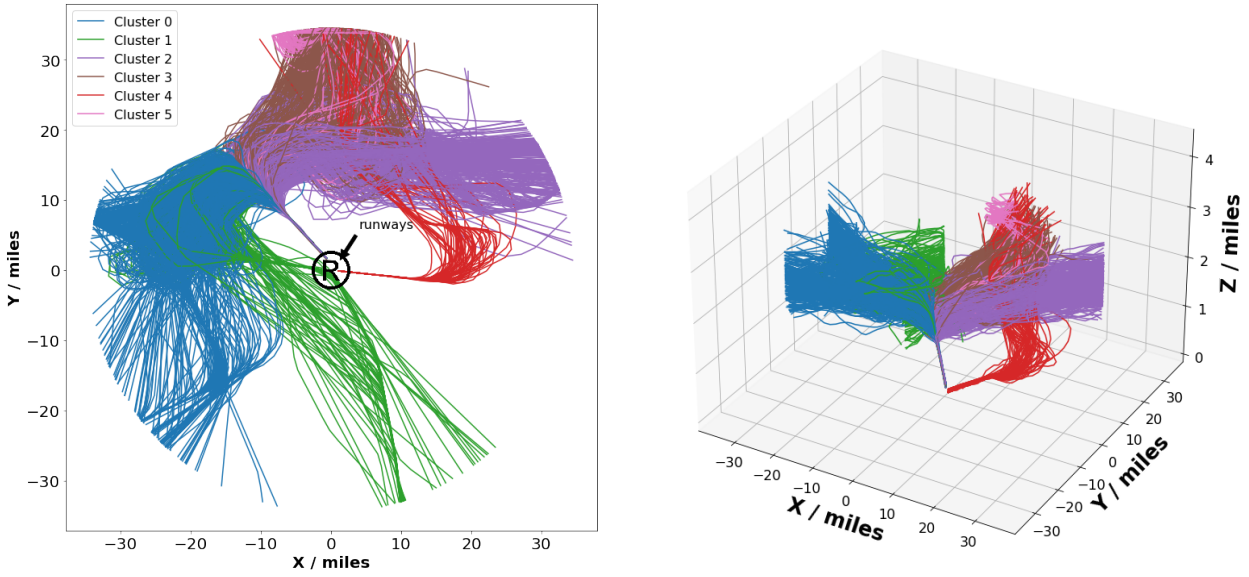
To facilitate a quantitative comparison between the two clustering methods, two key metrics have been defined as follows:

- **Density**: This metric is specific to a cluster and is calculated as the average distance between any two trajectories within the cluster. It reflects the closeness of the trajectories within one cluster.
- **Mean Density**: This metric pertains to the overall clustering result and is calculated as the **weighted average density** of all clusters (excluding the outliers) in the result set, based on the number of trajectories in each cluster.

It is reasonable to expect that a more reliable clustering would yield clusters with lower Densities, where the trajectories within the same cluster would be closer to each other. Consequently, the Mean Density of such a clustering would also have a lower value compared to a less reliable clustering method.

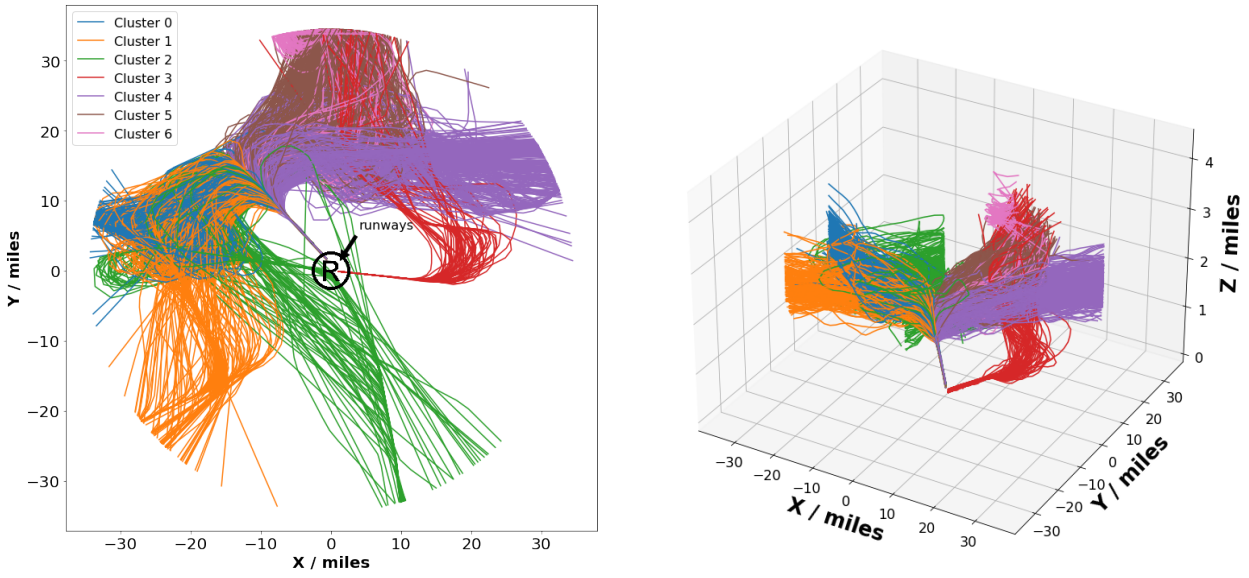
Taking Week 3 as an example, the outlier rates for OPTICS and DBSCAN clustering are 24.33% and 23.48% respectively, making the results from both algorithms comparable. Figure 6 illustrates the clusters obtained by DBSCAN, while Fig. 7 displays the clusters obtained by OPTICS, with both excluding the outlier trajectories.

The results indicate that both the OPTICS and DBSCAN algorithms have identified similar clusters, with OPTICS detecting an additional cluster compared to DBSCAN. Specifically, OPTICS recognizes Cluster 0 and Cluster 1 as separate clusters, while DBSCAN considers them as one. The epsilon distances of the trajectories, as depicted in Fig. 8, reveal that Cluster 0 exhibits a lower density compared to Cluster 1 (this observation is also supported by Fig. 7). This discrepancy highlights a limitation of DBSCAN in identifying clusters with varying densities due to its rigid



(a) 2D Results **(b) 3D Results**

Fig. 6 Clusters Obtained by DBSCAN for Week 3



(a) 2D Results **(b) 3D Results**

Fig. 7 Clusters Obtained by OPTICS for Week 3

neighborhood definition. In contrast, OPTICS, which does not depend on a fixed neighborhood, exhibits improved performance in identifying clusters with varying densities. As a result, it can achieve a more granular clustering process, ultimately producing clusters with reduced densities.

In terms of quantitative comparison, It is found that in Week 3, DBSCAN generates a Mean Density of 3.1378, while OPTICS produces a lower value of 2.4379. This finding suggests that OPTICS has generated a more reliable clustering result, which is consistent with our expectations based on the visualizations.

To further validate the reliability and consistency of OPTICS in clustering, a comprehensive comparison of the two clustering methods has been conducted across all weeks, excluding Week 4. The same approach for finding the parameters for the two clustering models has been strictly followed. The resulting outlier rates are shown in Table 2, and

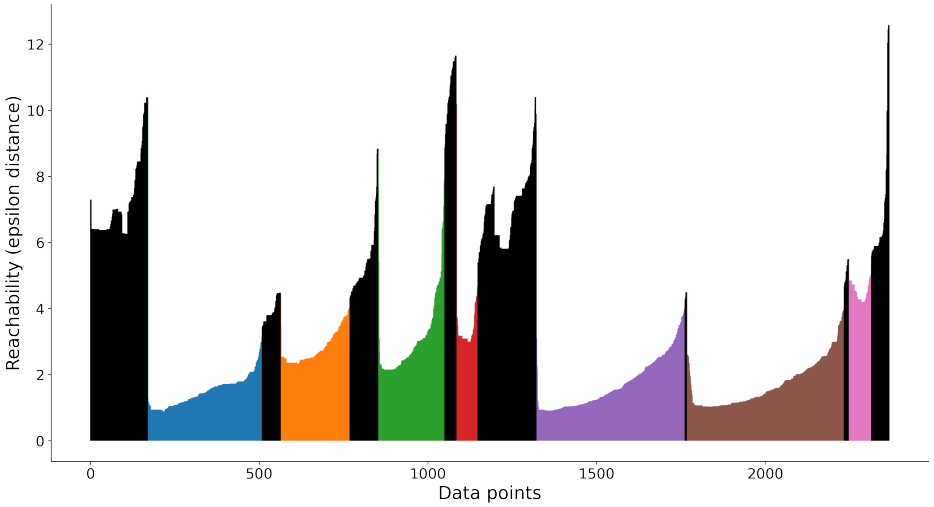


Fig. 8 Reachability Plot for Week 3

the clustering results are summarized in Fig. 9.

Table 2 Outlier Rate Across Weeks.

Week	OPTICS (%)	DBSCAN (%)	Week	OPTICS (%)	DBSCAN (%)
0	24.90	24.10	5	22.90	23.73
1	20.67	21.47	6	13.35	13.09
2	24.09	24.87	7	23.08	23.91
3	24.33	23.48	8	20.85	21.55

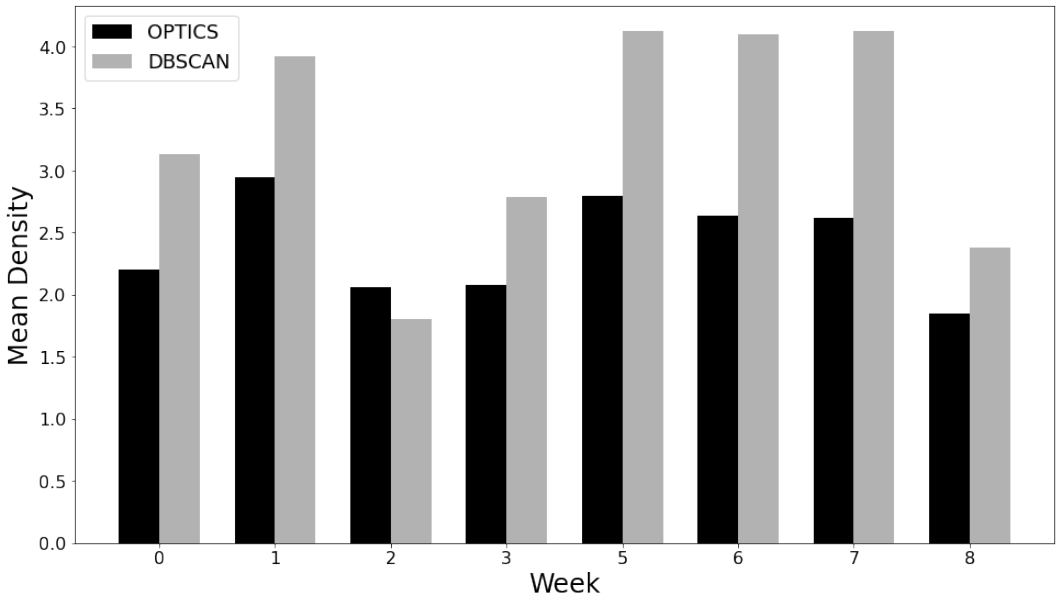


Fig. 9 Mean Density Across Weeks

The summary reveals a consistent pattern where OPTICS yields clustering results with lower Mean Density values compared to DBSCAN in 7 out of 8 weeks. The only exception is Week 2, where OPTICS exhibits a slightly higher Mean Density than DBSCAN. This consistent trend suggests that OPTICS has demonstrated a higher level of reliability in identifying traffic flows when compared to DBSCAN.

Given the more reliable clustering results produced by OPTICS and its requirement of only a single user-defined parameter compared to the two required by DBSCAN, we conclude that OPTICS is the more suitable clustering algorithm for identifying traffic flows in terminal airspace.

B. Classification

Optuna logs the hyperparameter tuning processes during the execution of the models. As an illustration, the hyperparameter tuning process for RF on Week 3, with an observation time of 180 seconds, is depicted in Fig. 10.

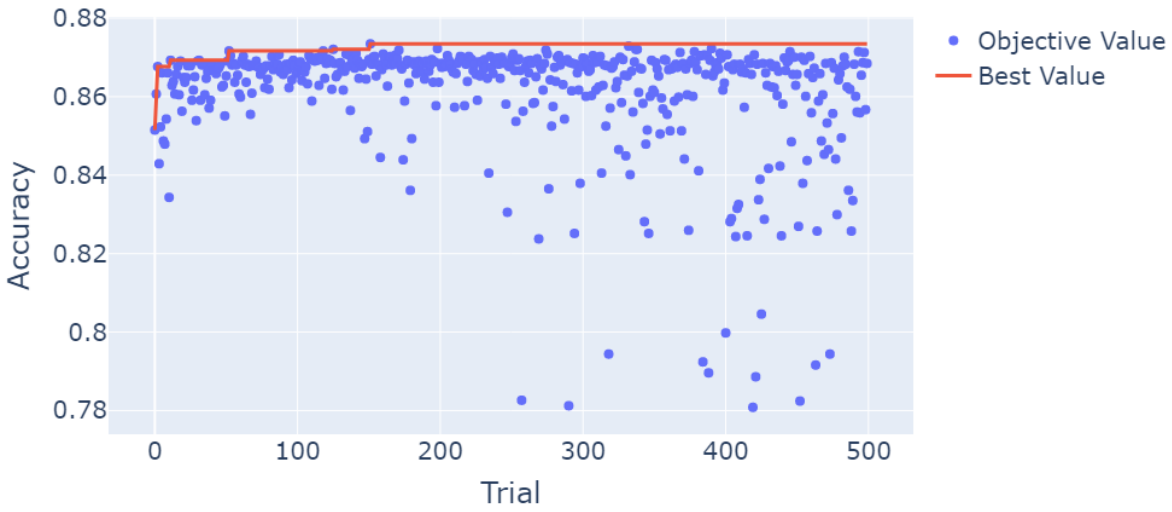


Fig. 10 Hyperparameter Tuning Logs - RF, Week 3, Observation Time 180 Seconds

The hyperparameter tuning process records the objective value, which represents the accuracy obtained by averaging the accuracy values of all cross-validation sets. This process consists of 500 trials, each corresponding to a specific combination of hyperparameters. While the majority of trials yield high accuracy values, some trials may result in lower values. This is expected due to the probabilistic nature of the Bayesian method, which allows exploration of suboptimal areas in the search space. The red curve in the graph represents the best value achieved among the completed trials. In the provided example, the best value initially increases within the first 200 iterations and then reaches a stable state.

Hyperparameter tuning is conducted for three classification model types. The final results are summarized in Table 11. To assess model performance, the accuracy values for each week, corresponding to the same observation time, are averaged to obtain an overall accuracy score.

After examining the accuracy results, a notable trend becomes evident, indicating an increase in overall accuracy as the observation time extends. This pattern aligns with expectations, as longer observation times provide the prediction model with more trajectory information, resulting in improved performance.

While our classification task does not exhibit a highly imbalanced input dataset, it is crucial to include additional classification metrics for a comprehensive evaluation of the model's performance. Therefore, we calculate and compare F1 score, also known as the harmonic mean of precision and recall [24]. The computation process for the F1 score is similar to our approach for accuracy, where we average the values over weeks. The results are presented in Table 3. Notably, the F1 score values demonstrate a behavior that closely aligns with the accuracy values, indicating the consistency between these two metrics in our classification models.

Another intriguing finding is that all three models achieve comparable levels of accuracy when subjected to the

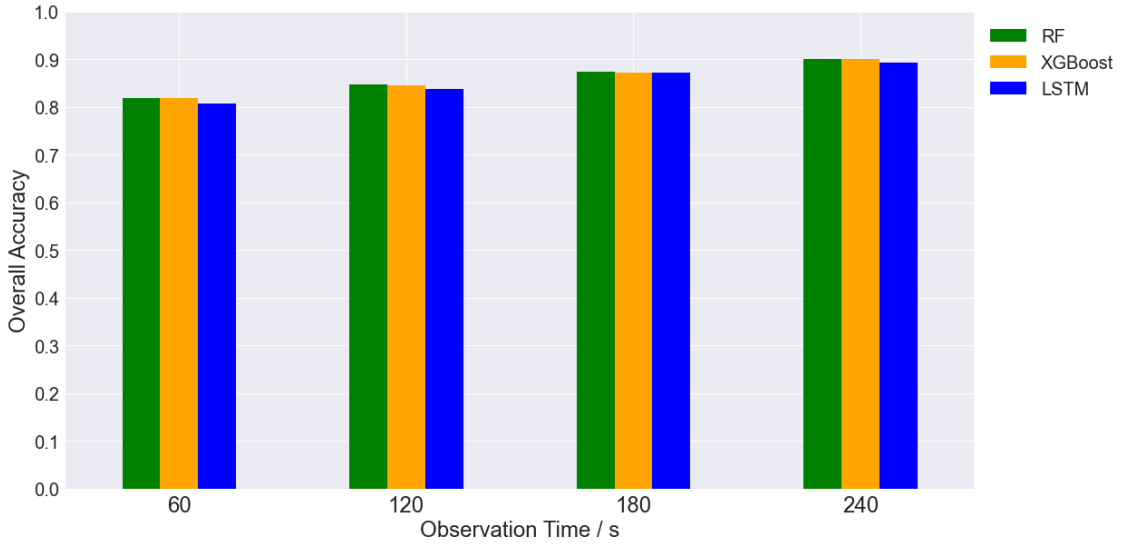


Fig. 11 Overall Accuracy Comparison Between Models

Table 3 Overall Accuracy Comparison

Observation	Model	Overall F1 Score	Observation	Model	Overall F1 Score
60	Random Forest	0.812	180	Random Forest	0.872
	XGBoost	0.814		XGBoost	0.871
	LSTM	0.809		LSTM	0.873
120	Random Forest	0.844	240	Random Forest	0.898
	XGBoost	0.842		XGBoost	0.899
	LSTM	0.836		LSTM	0.893

same observation times. This observation may initially seem surprising, considering that the LSTM, as a deep learning model, is expected to outperform ensemble models. However, a detailed discussion will provide an explanation.

A thorough evaluation of a model's classification performance involves analyzing the distribution of actual and predicted labels using a confusion matrix. This analysis enables the identification of specific areas that can be enhanced. Upon examining all confusion matrices obtained from models, it becomes apparent that the optimized models excel in accurately predicting and distinguishing between various traffic flows. However, they encounter challenges in effectively discerning regular traffic flows from outliers. To illustrate this phenomenon, Fig. 12 presents the confusion matrix for the classification results of Week 3 with an observation time of 180 seconds.

The confusion matrix is structured with rows representing truth classes and columns representing predicted classes. The matrix's diagonal elements signify the count of correctly classified samples, whereas the off-diagonal elements indicate misclassified samples. The majority of samples fall along the diagonal, indicating overall good performance of the model. In fact, the accuracy for this particular result set is 88%. Upon examining the misclassified samples, which are represented by the off-diagonal elements, it becomes apparent that a significant number of them are found in the first row or first column. This suggests that these samples have been misclassified as outliers when they are actually regular traffic flows, or vice versa.

Figure 13 provides deeper insights into this phenomenon, where all trajectories in the testing set are plotted using colors of their respective traffic flow numbers. This visualization demonstrates that trajectories for each regular traffic flow are distinct and easily distinguishable to human eyes. For instance, Cluster 1, depicted in orange, originates from the southwest and moves northeast. It is worth noting that identifying Cluster 3 and 5 might pose some challenges as they both originate from the north and head south. Importantly, outliers are scattered throughout the airspace, as they

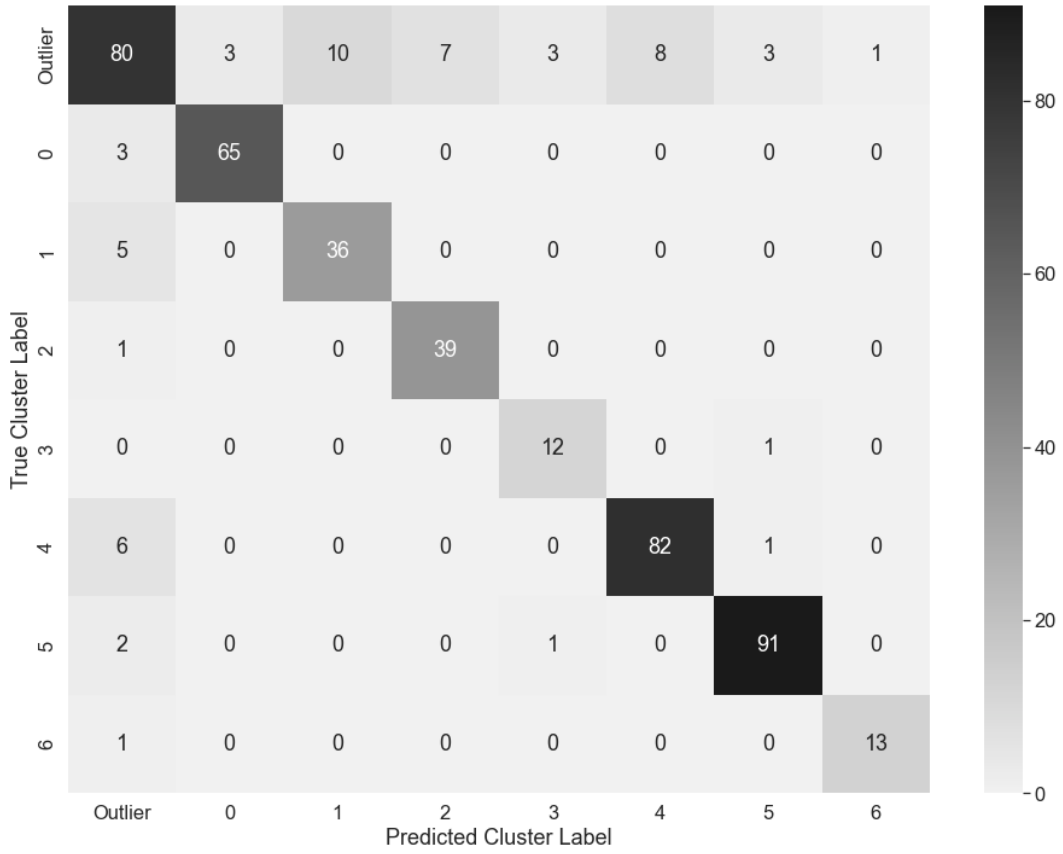


Fig. 12 Confusion Matrix - RF, Week 3, Observation Time 180 seconds

can emerge from any direction and move in any direction, making them blend in with regular traffic flows. This is a significant consequence of having incomplete trajectory data constrained by the observation time, which in this case is 180 seconds.

Consequently, differentiating between a trajectory segment containing regular traffic flow and an outlier, or vice versa, proves to be a challenging task even for humans. The Bayes error [25], also known as irreducible error or minimum achievable error, represents the lowest possible error rate that can be achieved for a given classification problem. It is a theoretical concept that reflects the best performance any classifier could attain, considering the statistical distribution of the data. While there is no defined method to compute the Bayes error for the current problem, the author believes that, based on the observations made, the bias and variance exhibited by the models approach the level of Bayes error.

Based on the previous discussion, there seems to be little variation in performance among the three models. However, notable distinctions emerge when considering their training costs. Table 4 summarizes the training times for the three model types, including cross-validation. Clearly, training an LSTM model requires significantly more time compared to training an ensemble model. Specifically, when comparing the two ensemble models, training a Random Forest (RF) model takes only about one-fifth of the time required for training an XGBoost model. This discrepancy arises from the RF model's ability to be constructed in parallel, while the XGBoost model is constructed sequentially. The shorter training time of the RF model leads to reduced durations for hyperparameter tuning and faster iterations during model development. Taking into account both model performance and training costs, the RF model is considered as the optimal selection for real-time classification of traffic flows in a given trajectory segment.

VII. Conclusion

The study of air traffic flows is significant in the aviation industry, as it enhances the understanding of traffic patterns and supports applications such as anomaly detection and conflict detection. In this research, we propose a methodology to identify traffic flows in terminal airspace and recognize traffic flows associated with real-time landing flights. We

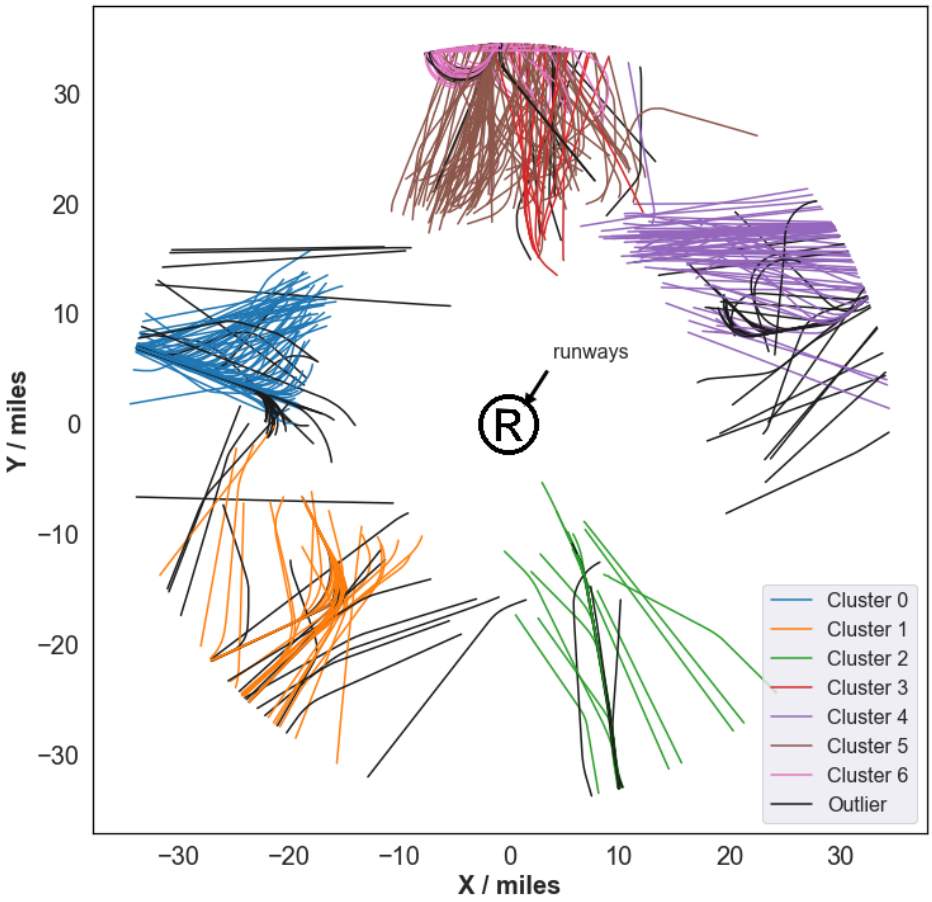


Fig. 13 Trajectories in the RF's Test Set for Week 3

Table 4 Training Time Comparison

Model Type	Approximate Training Time
RF	10 seconds
XGBoost	1 minute
LSTM	2 hours

acquire and preprocess raw ADS-B trajectory data from Zurich Airport as our primary data source. Our approach involves establishing clustering models using OPTICS and DBSCAN algorithms for traffic flow identification, as well as developing ensemble models (RF and XGBoost) and a LSTM model for traffic flow recognition. The obtained results demonstrate the superior reliability of OPTICS in identifying traffic flows compared to DBSCAN, making it the more suitable choice for this task. Furthermore, the findings demonstrate that ensemble models, particularly RF, achieve comparable performance to complex deep learning models in traffic flow recognition while requiring significantly less computational resource. Thus, RF emerges as the most suitable model for traffic flow recognition. By employing these methods and techniques, this study contributes to the advancement of traffic flow analysis and provides valuable insights for efficient and accurate traffic management in terminal airspace.

Acknowledgments

The authors would like to thank Dr. Tejas Puranik for generously sharing the original trajectory dataset used in this study and for engaging in valuable discussions with them.

References

[1] Olive, X., and Morio, J., "Trajectory clustering of air traffic flows around airports," *Aerospace Science and Technology*, Vol. 84, 2019, pp. 776–781.

[2] Madar, S., Puranik, T. G., and Mavris, D. N., "Application of Trajectory Clustering for Aircraft Conflict Detection," *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, IEEE, 2021, pp. 1–9.

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

[4] Murça, M., Delaura, R., Hansman, R., Jordan, R., Reynolds, T., and Balakrishnan, H., "Trajectory Clustering and Classification for Characterization of Air Traffic Flows," 2016. https://doi.org/10.2514/6.2016-3760.

[5] Murca, M. C. R., and Hansman, R. J., "Identification, characterization, and prediction of traffic flow patterns in multi-airport systems," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 20, No. 5, 2018, pp. 1683–1696.

[6] Kistan, T., Gardi, A., Sabatini, R., Ramasamy, S., and Batuwangala, E., "An evolutionary outlook of air traffic flow management techniques," *Progress in Aerospace Sciences*, Vol. 88, 2017, pp. 15–42. https://doi.org/https://doi.org/10.1016/j.paerosci.2016.10.001, URL https://www.sciencedirect.com/science/article/pii/S0376042116300458.

[7] Basora, L., Morio, J., and Mailhot, C., "A trajectory clustering framework to analyse air traffic flows," *7th SESAR Innovation Days*, 2017, pp. 1–8.

[8] Morris, B. T., and Trivedi, M. M., "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," *IEEE transactions on pattern analysis and machine intelligence*, Vol. 33, No. 11, 2011, pp. 2287–2301.

[9] *In-time Aviation Safety Management: Challenges and Research for an Evolving Aviation System*, National Academies Press, 2018.

[10] Rehm, F., "Clustering of flight tracks," *AIAA Infotech@ Aerospace 2010*, 2010, p. 3412.

[11] Gariel, M., Srivastava, A. N., and Feron, E., "Trajectory clustering and an application to airspace monitoring," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 12, No. 4, 2011, pp. 1511–1524.

[12] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al., "A density-based algorithm for discovering clusters in large spatial databases with noise." *kdd*, Vol. 96, 1996, pp. 226–231.

[13] Corrado, S. J., Puranik, T. G., Pinon, O. J., and Mavris, D. N., "Trajectory clustering within the terminal airspace utilizing a weighted distance function," *Multidisciplinary Digital Publishing Institute Proceedings*, Vol. 59, No. 1, 2020, p. 7.

[14] Bosson, C. S., and Nikoleris, T., "Supervised learning applied to air traffic trajectory classification," *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 1637.

[15] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J., "OPTICS: Ordering points to identify the clustering structure," *ACM Sigmod record*, Vol. 28, No. 2, 1999, pp. 49–60.

[16] Breiman, L., "Random forests," *Machine learning*, Vol. 45, No. 1, 2001, pp. 5–32.

[17] Chen, T., and Guestrin, C., "Xgboost: A scalable tree boosting system," *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[18] Hochreiter, S., and Schmidhuber, J., "Long short-term memory," *Neural computation*, Vol. 9, No. 8, 1997, pp. 1735–1780.

[19] Olive, X., "Traffic, a toolbox for processing and analysing air traffic data," *Journal of Open Source Software*, Vol. 4, No. 39, 2019, pp. 1518–1.

[20] 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," *the Journal of machine Learning research*, Vol. 12, 2011, pp. 2825–2830.

[21] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, Vol. 32, 2019.

[22] Snoek, J., Larochelle, H., and Adams, R. P., "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, Vol. 25, 2012.

[23] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M., "Optuna: A next-generation hyperparameter optimization framework," *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[24] Goutte, C., and Gaussier, E., "A probabilistic interpretation of precision, recall and F-score, with implication for evaluation," *Advances in Information Retrieval: 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005. Proceedings 27*, Springer, 2005, pp. 345–359.

[25] Tumer, K., and Ghosh, J., "Estimating the Bayes error rate through classifier combining," *Proceedings of 13th international conference on pattern recognition*, Vol. 2, IEEE, 1996, pp. 695–699.