electronics

MDPI

Article

Detecting GPS Interference Using Automatic Dependent Surveillance-Broadcast Data

Akshay Ram Ramchandra [1,†], Anton Skurdal [1,†], Prakash Ranganathan [1,*,†] and William Semke [2,†]

[1] School of Electrical Engineering & Computer Science, University of North Dakota, Grand Forks, ND 58202, USA; akshay.ramchandra@und.edu (A.R.R.); anton.skurdal@und.edu (A.S.)
[2] Department of Mechanical Engineering, University of North Dakota, Grand Forks, ND 58202, USA; william.semke@und.edu
* Correspondence: prakash.ranganathan@und.edu
† These authors contributed equally to this work.

Abstract: This paper investigates the detection of Global Positioning System (GPS) interference during the Dallas Fort Worth (DFW) event from 17 to 19 October 2022, utilizing various machine learning (ML) models. The study examines the effectiveness of several ML models, including neural networks (NN), tree-based models, regression-based models, Bayesian classifiers, distance-based models, and stochastic classifiers, in identifying GPS interference. A simulated training signature was created with 180,000 data points, of which 25,792 were modeled as positive samples indicating GPS interference. Preliminary results reveal that the Multi-Layer Perceptron (MLP) model outperformed others, achieving a 99.8% True Positive Rate (TPR). Additionally, permutation feature importance was utilized to understand how model feature prioritization impacts the detection outcomes. Given the increasing frequency of GPS interference, these findings underscore the critical importance of ML techniques in detecting GPS interference patterns in Automatic Dependent Surveillance-Broadcast (ADS-B) data.

Keywords: GPS interference; automatic dependent surveillance-broadcast; anomaly detection; machine learning; data processing; ADS-B

Citation: Ramchandra, A.R.; Skurdal, A.; Ranganathan, P.; Semke, W. Detecting GPS Interference Using Automatic Dependent Surveillance-Broadcast Data. *Electronics* **2024**, *13*, 3145. https://doi.org/10.3390/electronics13163145

Academic Editor: Srinivas (Srini) Sampalli

Received: 25 June 2024
Revised: 25 July 2024
Accepted: 5 August 2024
Published: 8 August 2024

1. Introduction

Prior to the advent of radio navigation during World War II [1], pilots relied heavily on visual cues during the day (road maps) and on searchlights and bonfires at night for navigation. The introduction of radio navigation marked a significant advancement in aviation by allowing aircraft to determine their position more accurately. Ground stations used the very high-frequency (VHF) band to broadcast position-finding signals that aircraft latch on to determine their location. However, in areas with sparse ground station coverage, aircraft were forced to make deviations to flight plans to remain within the range of these broadcasts. At its peak, the United States had over 1000 ground stations dedicated to this purpose. The advent of GPS was truly revolutionary to aircraft navigation as it became fully operational and available for civil use by 1995. This satellite-based navigation system allows aircraft to obtain precise three-dimensional positioning information anywhere on Earth, eliminating the need for ground-based navigation aids and significantly enhancing the accuracy and reliability of aircraft navigation. GPS has become a cornerstone of modern aviation, with both civil and military sectors heavily dependent on it for navigation purposes [2–4]. The widespread adoption of GPS is evident in the staggering number of receivers in use—by 2019, the United States alone reported over 900 million GPS devices [5]. This extensive integration of GPS technology into aviation systems highlights its critical role in ensuring safe and efficient air travel while also underscoring the potential vulnerabilities that could arise from disruptions to GPS signals. The GPS is susceptible to malicious and unintentional interference because the signals from GPS satellites are extremely faint [6].

According to the International Air Transport Association (IATA), the average number of GPS signal-loss events was 24 per 1000 flights before May 2023. This number spiked to 40 events per 1000 flights in August 2023 before dropping slightly to 34 per 1000 flights later that year. The majority of these incidents occurred in Eastern Europe and the Middle East due to conflicts in Ukraine and Syria, where GPS jamming and spoofing are used as defense mechanisms against drones [7]. In August 2018, a civilian aircraft with passengers navigating in restricted visibility veered off course due to GPS interference and was saved at the last minute by the intervention of ATC, avoiding a mountain collision [1]. In early 2022, Denver Airport experienced an interference event lasting over 33 h, affecting aircraft in an 8000 square-mile area around Denver at altitudes of up to 14,000 feet [8]. Another episode at DFW lasted almost 48 h and caused an entire runway to shut down, with several flights diverted to other airports [9]. Additionally, interference can significantly impact devices like Unmanned Aerial Vehicles (UAVs), affecting their GPS, compass, and other central control modules [10,11]. In December 2012, a passenger jet near Reno, Nevada, veered 10 miles off course due to military GPS jamming, necessitating air traffic control intervention to prevent a collision. The U.S. military regularly conducts GPS jamming tests, impacting civilian aviation [1]. In 2017, 173 incidents of GPS interference were reported over six months, affecting various aircraft types in Southern California, Nevada, Utah, Arizona, and the Pacific Ocean. These incidents prompted the FAA to issue Notices to Airmen (NOTAM) to inform pilots of GPS disruptions during military tests, although this has caused anxiety among civilian aviation professionals [12]. To address these issues, the FAA is upgrading to the Next-Generation Air Transportation System (NextGen), which will improve aviation safety by utilizing satellite-based navigation systems [6]. The ADS-B is the cornerstone of this initiative.

1.1. ADS-B and ADS-B Databases

The ADS-B system works by periodically broadcasting aircraft state parameters without operator intervention by using other navigational systems to gather position, altitude, and velocity information [13–15]. The data are available to anyone with appropriate receiving equipment and aim to enhance situational awareness for pilots and Air Traffic Controllers alike. The information transmitted by the ADS-B (segregated by message type) is outlined in Table 1. ADS-B version 2 broadcasts state parameters and operational status messages that indicate the accuracy or quality of the transmitted positional information.

Table 1. Content and characteristics of various ADS-B message types [16].

Category	Transmitted Information
Identification Messages	Callsign and Wake Vortex Category
Airborne Position	Latitude, Longitude, and Altitude
Surface Position	Latitude, Longitude, velocity, and aircraft heading
Airborne Velocity	Vertical rate, Altitude (Barometric and GNSS), Ground Speed, and Air Speed
Operational Status	Capacity class, Operational mode, ADS-B version, NIC, NAC–position, and SIL supplement

Operational status parameters include uncertainty metrics. The Navigation Integrity Category (NIC) indicates position accuracy, with higher values denoting greater precision. NIC values range from 0 to 11, reflecting the containment radius of the aircraft position [16]. NIC superseded the earlier Navigational Uncertainty Category (NUCp) with the introduction of ADS-B Version 1. NIC values indicate the containment radius of the aircraft's position. NIC values and containment radius are inversely proportional; a higher NIC indicates a lower containment radius, while a lower NIC indicates a higher containment radius. A value of 11 has the least containment radius of about 7.5 m, while a value of 0 indicates an infinite containment radius, indicating a complete loss of position. The FAA considers NIC values of 7 and above as reliable positions [13,17].

Since 2013, the OpenSky Network has operated as a non-profit, crowd-sourced initiative that collects aviation data globally using off-the-shelf ADS-B receivers. The network

processes and stores this information in a central database. The collected data included aircraft positional details (both airborne and surface), identification, velocity, operational status, and uncertainty metrics. This information is transmitted by ADS-B-equipped aircraft when within range of the Opensky network's volunteer-operated sensors. OpenSky Network data has been used in several applications like ADS-B error and fault diagnosis, aircraft performance evaluation, ADS-B data validation, aircraft position multilateration, security analysis, and air traffic modeling [18–22]. The Open Sky Network's historical database is available for research and non-commercial use by applying for an educational account at their website [23]. For these reasons, the OpenSky Network is chosen as the ADS-B data source. The OpenSky Network uses Apache Impala, a popular Hadoop Distributed File System (HDFS) database that is capable of handling the large volumes of data generated by the ADS-B networks [24].

1.2. Current-State-of-Research

Current research on detecting GPS interference can be grouped into data-driven approaches, satellite-based techniques, and receiver-based methods. Murrian et al. [25] use Low Earth Orbiting (LEO) to detect the presence of interference from ground-based sources by using a Software Defined Radio (SDR) that listens for signals on the L1 and L2 GPS bands. The idea is that GPS signals originate from outer space and are faint. However, interference sources are much stronger and originate from the Earth. Such instances were recorded and localized using the Doppler shift. SWEPOS, a network of satellites, is used to monitor and detect Global Navigation Satellite Systems (GNSS) interference by analyzing the historic Signal-to-noise ratio (SNR) of different GNSS, including GPS, GLONASS, etc. The historic SNR characteristics of multiple satellites, in combination with statistical methods, help differentiate RFI sources. The detection capabilities in both simulated and real-world scenarios are shown by Abraha et al. [26].

Methods also focus on using the GPS receiver to detect interference. O'Mahony et al. [27] used received in-phase and quadrature samples and employed an ML-based approach to detect interference in Edge devices, using simulated SDR data and is useable by resource-constraint edge devices like the Raspberry Pi. Other methods, like Sun et al.'s [28], use a re-arranged Wavelet–Hough transform to detect common interference signals like sweep and continuous waves. While previous approaches focused on GPS interference sources from Earth, Patil et al. [29] investigated space-based GNSS interference with a network of 43 frequency receivers in the US and Europe and identified a power spike at 1268.52 MHz, which was traceable to satellites.

ADS-B data-driven approaches rely on data from ADS-B/ADS-B databases to analyze and find patterns indicative of GPS interference. Using a jammer and aircraft on the ground, Lukevs et al. [30] recorded the ADS-B transmission of an aircraft and found that NACp dropped from an acceptable value of 9 to below 7. Liu et al. [31] analyzed pilot reports of interrupted GNSS service and recorded ADS-B messages from a test flight during a GPS interference exercise conducted at Edwards Air Force Base. The main finding was similar to Lukevs et al.'s: a combination of low NIC values and ADS-B dropouts is typical of a GPS interference event. The authors also found that as the jammer was moved farther from the aircraft, the NACp gradually recovered and stabilized at a distance of 275 m (902.2 Feet). The main finding of this work was that the NACp values dropped, and there were gaps in the transmission of ADS-B messages when the aircraft's GPS was affected by an RF interference source. Ala et al. [32] builds on the finding that there is a gap in the continuous transmission of ADS-B messages due to GPS interference. A moving average of NACp is calculated and a threshold of 0.135 (moving average of NACp) is established. Readings that exceed this threshold suggest potential GPS jamming. The loss of messages is leveraged to find the likely location of the jammer by Jonavs et al. [33]. First, they correlate and rule out other causes of interference, such as military testing, constellation or satellite failure, and space weather, using relevant data sources. Once these are ruled out, they assume RF interference as the cause and use the Friis transmission equation to estimate

the jammer's position, assuming that the start of the gap in message reception is the point closest to the Jammer. Lui et al. [34] has created a real-time monitoring system capable of updating probabilities every 30 s and is capable of localizing the location of the jammer within 20 min of interference onset, using real-time ADS-B data. The authors divide the airspace into sections using the Bayesian updating algorithm, updating the probability of GNSS interference every 30 s based on NIC values. The jammer's location and power are estimated by minimizing the difference between estimated and measured jamming power using Friis's formula, refined interactively with the Gauss-Newton method. Research in this category is consistent in finding the pattern of GPS interference, which is a gap in time with a drop-in NIC around the gap.

A summary of the findings from the literature review is provided in Table 2. Satellite and receiver-based techniques have shown effectiveness in localizing interference, leveraging signal characteristics, and Doppler shifts. Receiver-based techniques that leverage ML, and signal processing may be costly, invasive (require modification to existing equipment), and necessitate testing before they can be implemented in critical applications, which may not always be practical for widespread use. In contrast, data-driven approaches, particularly those using open-source databases, offer a more cost-effective and non-invasive alternative. The analysis of ADS-B data has resulted in the identification of a pattern, which is a gap in ADS-B data, with a drop in NIC from above seven to below seven on either side or both sides of the gap by multiple studies [25,30,32,33,35]. Our proposed methodology focuses on the use of ML to identify the GPS interference pattern in ADS-B data. Existing methods use complex convolutional neural networks in addition to conventional methods like logistic regression to detect GPS interference using ML [36]. Our methodology builds on these methods and uses simpler, conventional algorithms to detect GPS interference. We focus on doing so in an efficient manner that may be suitable for real-time application. This work will focus on answering the following research questions:

1. What are the characteristics of NIC during GPS interference events, and can the patterns described in the existing literature be observed in actual GPS interference incidents?
2. Which ML algorithm is able to detect the GPS interference pattern accurately?
3. Which algorithm is computationally inexpensive?

This paper is organized into the following sections: The method section explains how data was acquired and analyzed to understand the properties of NIC during a GPS interference event. It details the challenges faced during data acquisition, like server timeouts and large file sizes, and how they were overcome using a Python script. The creation of synthetic ADS-B data to reflect real-world conditions and the training process of ML models to detect GPS interference patterns, aiming for a balance between computational efficiency and prediction accuracy, is explained. Results: This section details the outcomes of the model training process and results, providing insights into model performance during the training phase, variance testing, real-world applicability, and computational efficiency. Conclusion: This section concludes the investigation into GPS interference through ADS-B data; it also documents future work.

Table 2. Summary of the literature on GNSS interference detection.

Ref	Classification	Methodology	Key Findings
Murrian 2021 [25]	Satellite-based techniques	LEO and SDR listening on L1 and L2 GPS bands	Detected and localized ground-based interference using Doppler shift
Abraha 2024 [26]	Satellite-based techniques	Historic SNR characteristics of multiple satellites	A strong correlation between SNR anomalies and specific interference events
Patil 2023 [29]	Receiver-based techniques	Network of 43 frequency receivers	Identified space-based GNSS interference, traced to Beidou 3S-M1S satellite

Table 2. *Cont.*

Ref	Classification	Methodology	Key Findings
O'Mahony 2021 [27]	Receiver-based methods	ML-based approach with edge devices	Detected interference and jamming attacks using in-phase and quadrature samples
Sun 2021 [28]	Receiver-based methods	Wavelet–Hough transform	Detected common interference signals like sweep and continuous waves
Lukevs 2020 [30]	Data-driven approaches	ADS-B data analysis	Found significant role of jammer position on NACp values
Liu 2020 [31]	Data-driven approaches	Pilot reports and test flights	Identified ADS-B message dropouts and low NIC values during GPS interference
Ala 2019 [32]	Data-driven approaches	Analysis of ADS-B NACp values	Monitored values followed a semi-normal distribution
Jonavs 2019 [33]	Data-driven approaches	Triangulation using Friis transmission equation	Identified possible RFI source
Liu 2022 [34]	Data-driven approaches	Real-time monitoring system	Localized jammer within 20 min using Bayesian updating algorithm

2. Materials and Methods

Our methodology revolves around the innovative use of machine learning to identify GPS interference patterns in ADS-B data. Figure 1 depicts an overview of the methodology. GPS interference, as it appears in ADS-B data, is characterized by a gap in continuous messages and a drop in NIC on either or both sides of the gap (Section 1.2). The ADS-B data of a GPS interference event that led to the issue of FAA advisories regarding GPS unreliability [9] is acquired from the OpenSky Network and analyzed with NIC as the primary parameter. Insights from the analysis are used to create a data generator that can produce data closely resembling the real-world characteristics of NIC and message reception time. In total, 600 aircrafts that contain the GPS interference pattern are sifted to create a validation set. ML models are trained on the data from the generator and validated on the 600 positive samples, and explainable AI techniques are used to understand variations in the performance of the models. A strict selection methodology is applied to select the optimal model that can strike a balance between prediction speed and accuracy. The methodology is divided into 3 subsections. Section 2.1 describes the challenges in the data acquisition process and how it was overcome by developing a Python script. Section 2.2 describes the analysis of the acquired data, giving insight into the properties of NIC during a GPS interference event. The process of creating a set of 600 positive samples for the validation of the trained ML model is described in this subsection. Finally, Section 2.3, describes the methodology used to train the ML models.

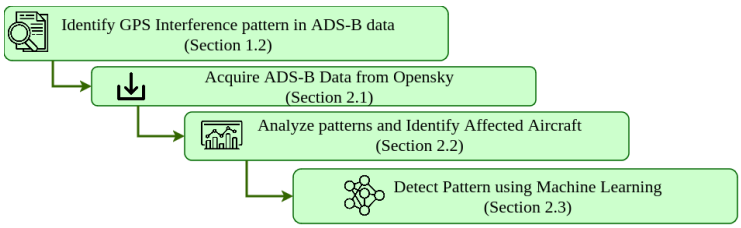

Figure 1. Multi-step GPS integrity interference section process.

2.1. Data Aquisition

Data for the analysis was sourced from the OpenSky Network's central impala database [24] that houses data in 15 different tables based on message type and data version. Access to the data is facilitated exclusively through Secure Shell (SSH) and requires data retrieval using Structured Query Language (SQL) queries. The resultant data are presented in the terminal output, which needs to be saved to a file and parsed using

regular expressions to obtain a usable Comma Separated Value (CSV) file. This method is well documented in the Open Sky Network's historical database documentation [23]. The state_vectorsdata_4 and operational_status_data4 are the two tables that are of interest to us. The former contains aircraft state parameters like time, position, altitude, velocity, heading, etc., while the latter contains uncertainty metrics like NIC, NAC, etc. The common column among both these tables is ICAO24, which uniquely identifies an aircraft and time.

The operational status data encapsulates time as a range (min and max time), contrasting singular time steps in the state vectors table, complicating data joins and increasing computational demand. The complexity of queries necessary for joins, coupled with server timeouts due to the large size and retrieval time presented significant challenges. These issues resulted in restarting downloads from scratch, which wasted much time. Further, the single file data dump posed yet another issue that necessitated special measures due to its large size. To overcome these challenges, a custom Python script was developed to facilitate data retrieval. Figure 2 visualizes this process, where the red rectangle is the user input (SQL Queries), the purple block denotes the user's environment, and the blue block signifies the OpenSky Network's server. The script takes the SQL query as input, connects to the OpenSky Network, and executes the query to retrieve data from the state vectors table. Upon obtaining the state vector data, the script extracts the unique ICAO IDs and queries the operational status messages table using the ICAO ID and time. This method of querying helps break down the data into smaller consumable chunks filtered by unique aircraft, and time helps track download progress and resume the download from the last data point downloaded instead of restarting downloads from the beginning. This approach helps overcome the server timeout issue and not waste time/computational resources to restart large downloads. Breaking down the queries by individual aircraft helped chunk the data into smaller and processable sizes. The logic used to join time which is described as a single time stamp in the state vectors and a range in operational status is described in Algorithm 1. The script handles each aircraft independently of another, allowing us to leverage multi-threading to speed up the join process. We currently utilize 5 threads that enable the processing of 5 aircraft in parallel. This method accelerates data processing and ensures data integrity and completeness. This code is able to resume downloads from the last data point (when downloading the operational status messages), saving time when server timeouts occur, joining the state vectors, and operational status messages table, and produce data in smaller consumable chunks that are easier to process. The script is available online in our DECSResearch GitHub repo [37]. This program was used to query data about the DFW Interference event, which is described and analyzed in the following subsection.

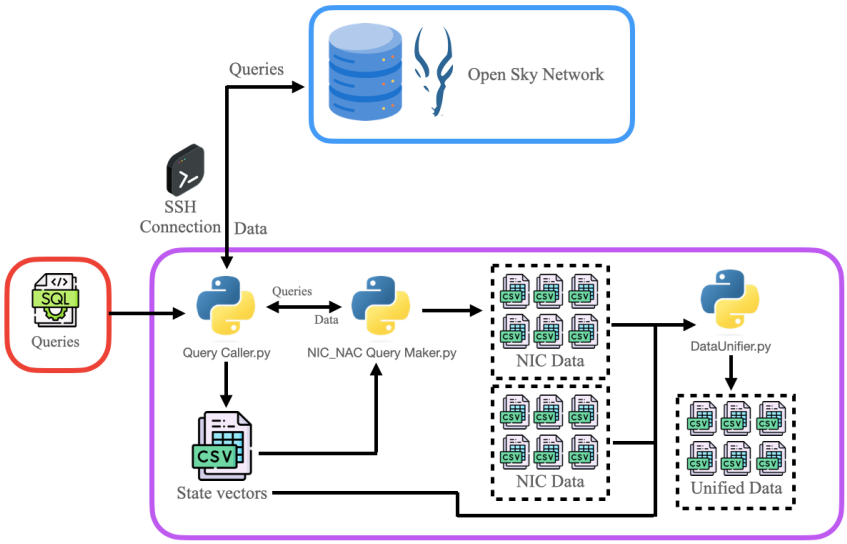

Figure 2. Python scripts to facilitate data downloads from the OpenSky Network.

Algorithm 1 Time Handling Logic to join NIC Values to State Vectors

Require: df: DataFrame, nic_df_path: Path to NIC files, nic_added_path: Path for saving updated DataFrames
Ensure: DataFrames with NIC values and message counts, considering specific time-matching criteria, are saved to the specified path

procedure ADD_NIC(df, nic_df_path, nic_added_path)
    Initialize $pool\_size \leftarrow 5$
    Create a thread pool with $pool\_size$
    Extract unique $icao24$ codes from df
    for each $unique\_icao$ in unique $icao24$ codes **do**
        Apply $nic\_added\_inner\_fun$ asynchronously with arguments ($unique\_icao$, df, nic_df_path, nic_added_path)
    end for
    : Wait for all tasks in the pool to be complete
end procedure
procedure NIC_ADDED_INNER($unique\_icao$, df, nic_df_path, nic_added_path)
    Filter df for $unique\_icao$, resulting in ic_df
    Load corresponding NIC DataFrame from nic_df_path
    for each $lastposupdate$ in ic_df **do**
        Attempt to match $lastposupdate$ directly with $maxtime$ or $mintime$ in NIC DataFrame
        if direct match found **then**
            Use NIC value and message count from matched row
        else
            Adjust $lastposupdate$ by rounding or converting to integer
            Retry matching with adjusted $lastposupdate$ values
            if match found with adjusted values **then**
                Use NIC value and message count from matched row
            else
                Determine the closest $maxtime$ or $mintime$ within a 2-second threshold
                if a close enough time is found **then**
                    Use NIC value and message count from the closest time row
                else
                    Set NIC value and message count to NaN
                end if
            end if
        end if
    end for
    Add NIC values and message counts to ic_df
    Save updated ic_df to nic_added_path
end procedure

2.2. GPS Interference Analysis

The initial 28 hours of the DFW GPS interference event are analyzed. The OpenSky Network's historical database was queried in four hours chunks to obtain data from 17 October 2022, 20:00:00 to 18 October 2022, 23:59:59, for a radius of 74.08 km (40 Nautical Miles (NM)) around DFW airport. A total of 5,747,931 data points and 2559 unique aircraft were present in the entire dataset. The queries used to obtain the data are available in our GitHub repository's [37] 'queries' directory. These data are analyzed with NIC as the primary parameter. NIC is compared with other aircraft state parameters, such as altitude, change in NIC, and aircraft heading. The findings from this analysis are detailed below:

1. **Altitude-related interference.** Numerous instances of NIC dropping to 0 at altitudes as high as 9000 to 15,000 m was observed. This is unusual for airborne aircraft, which usually have a clear view of the satellites above the horizon. Figure 3 is a violin plot that visualizes this finding. The *X*-axis is the NIC, which ranges from 0 to 11 (also depicted in color), and the *Y*-axis is the aircraft's altitude.

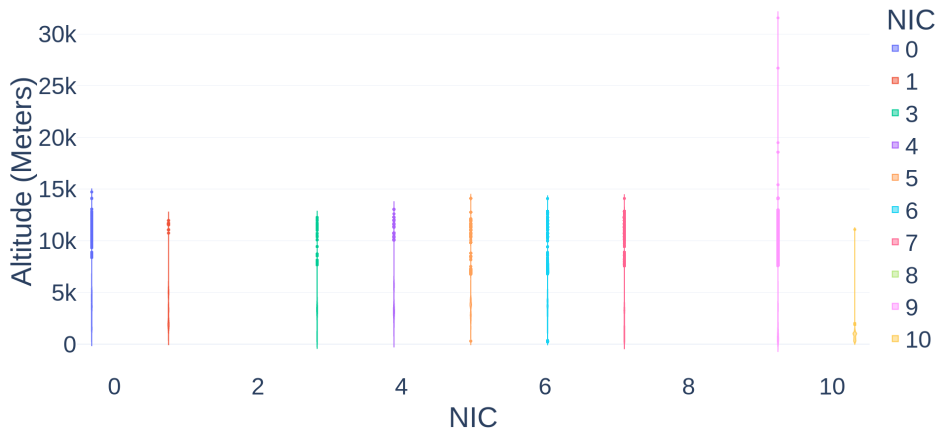

Figure 3. Veiolin plot of altitude in meters and NIC in color.

2. **Change in NIC values.** The change in NIC was studied to understand when it drops to zero, what value it begins from, and to what values it recovers. It was found that the most typical fluctuation in NIC was from 9 to 0 and back to 9, followed by 6 to 0 and a recovery back to 6. This is shown as a histogram in Figure 4. The *X*-axis of the plot is the change in NIC; for example, a value of '7_*TO*_6' represents an initial value of 7, after which the NIC dropped to 0 and recovered a value of 6. The *Y*-axis is the count of the number of times each of these instances occurred in the entire dataset.

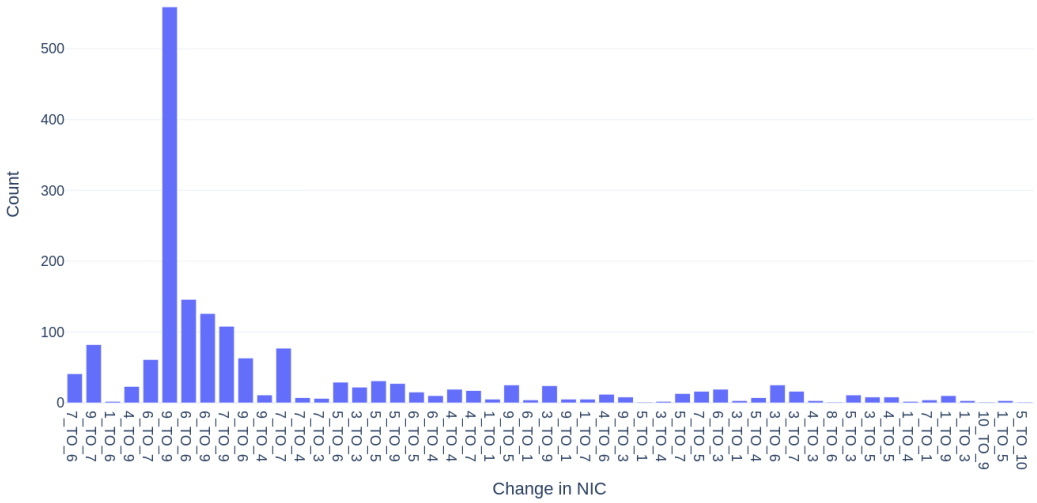

Figure 4. Count plot of change in NIC by category (7_TO_6 means NIC changed from 7 to 0 and recovered to 6).

3. **Aircraft heading.** A notable pattern between aircraft heading and Navigation Integrity Category (NIC) values was found. As illustrated in Figures 5 and 6, aircraft approaching the airport from the northeast (heading southwest) showed a higher occurrence of NIC 0 values. This is particularly striking given the relatively low number of aircraft flying in this direction. Figure 6 displays a concentration of NIC 0 data points in the southwestern quadrant, while Figure 5 confirms that this is not due to higher traffic volume in this direction.

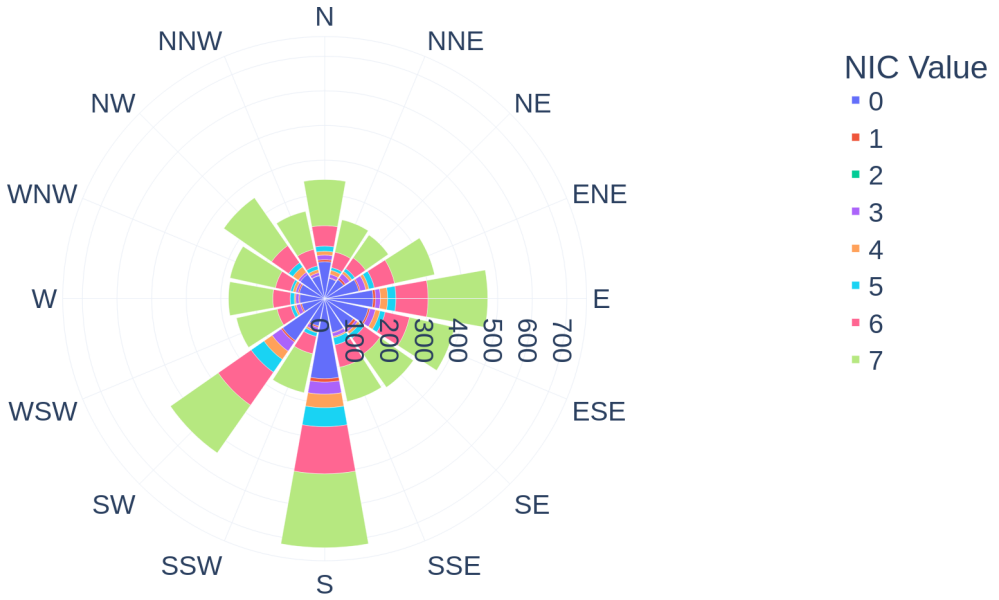

Figure 5. Polar bar chart of the distribution of number of aircraft, by heading with NIC as color.

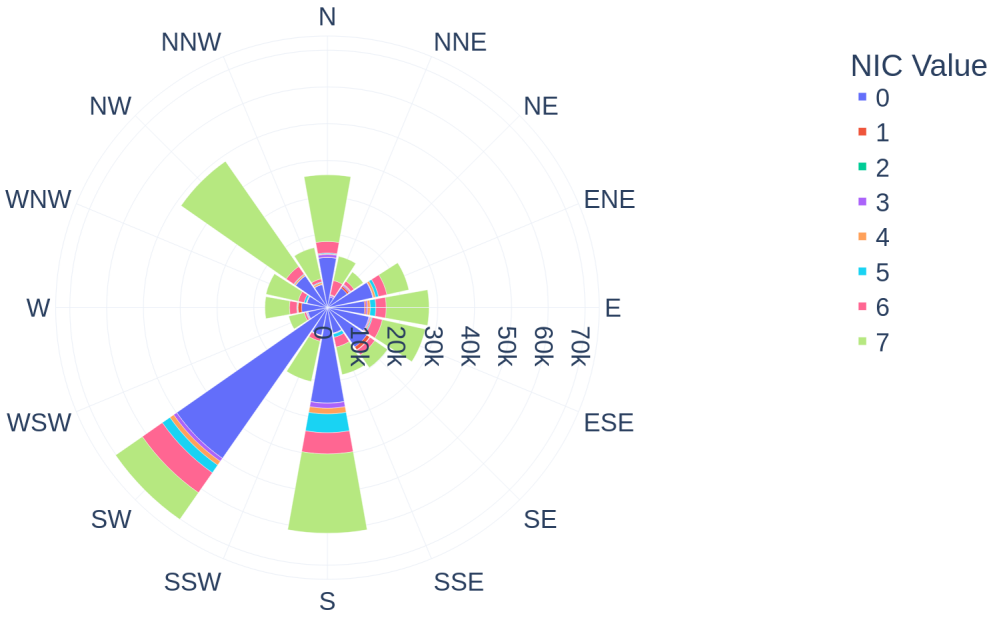

Figure 6. Polar bar chart of the distribution of number of datapoints, by heading with NIC as color.

4. **Individual aircraft analysis.** Individual aircraft trajectories were visualized iteratively to identify aircraft that contained the GPS interference pattern (a gap in continuous ADS-B transmission and a drop in NIC on both or either side of the gap). A basic visualization tool was created that presented two- and three-dimensional plots of the aircraft's trajectories and prompted user input to create a collection of positive samples. These sets of aircraft are confirmed to contain the GPS interference pattern and will serve as the validation set for testing model performance. A sample two-dimensional scatter plot of the latitude and longitude of an aircraft with NIC as color is visualized in Figure 7. The red annotations on the plot show the GPS interference pattern that is expected of a positive sample (contains GPS interference). Figure 8 is a three-dimensional scatter plot that shows the trajectory of another aircraft that contains the GPS interference pattern. Aircraft with similar trajectories were sifted, and a total of 600 such aircraft trajectories were collected to create the validation set for the machine-learning model.

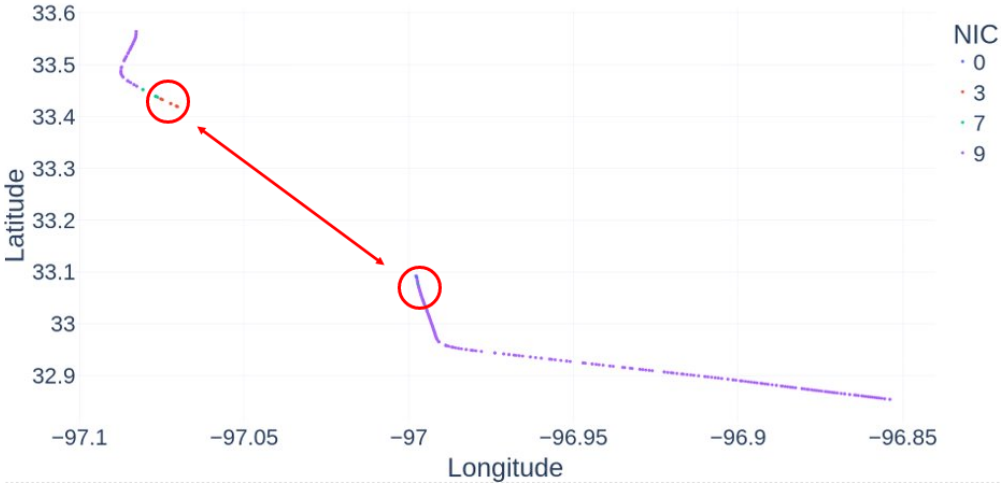

Figure 7. 2-Dimentional Visualization: Single aircraft visualization with NIC as color (GPS interference pattern highlighted in red).

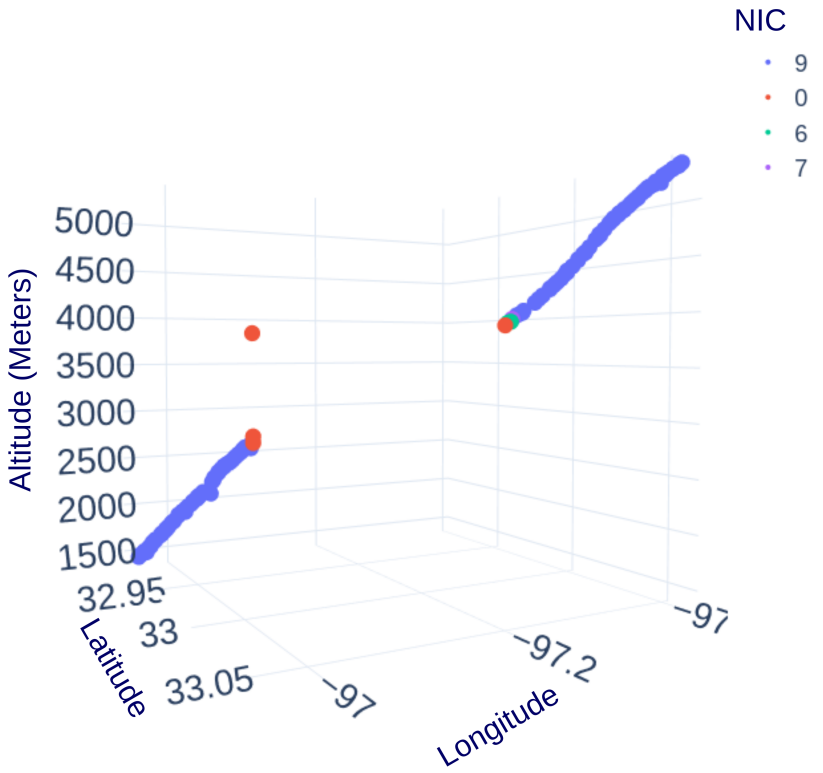

Figure 8. 3-Dimensional Visualization of aircraft trajectory, showing ADS-B characteristics of GPS interference.

5. **NIC and time properties.** Time and NIC have been identified as the key parameters to detect GPS interference; hence, to simulate data that is close to the real world, these two features are analyzed in depth. A summary of the probabilities is provided in Table 3. An NIC of 9 is the most typical value with a probability of 0.9016 while an NIC of 11 is completely absent, which could be attributed to the fact that it requires the addition of Satellite-Based Augmentation System (SBAS) to the GPS, which enhances the accuracy and reliability of GNSS signals by using a network of ground stations and geostationary satellites [38,39]. The time gaps in the dataset ranged from 1 to 50,000 s; the large range of values is likely caused by geographic filtering, which

resulted in aircraft entering and leaving the region of interest, introducing substantial variability in the time intervals.

Table 3. NIC probabilities extracted for NIC ≥ 7 (normal data) from the data acquired from OpenSky.

NIC Value	Probability
7	0.03116
8	0.0010
9	0.9016
10	0.0561
11	0

2.3. GPS Interference Detection Using ML

The GPS interference pattern identified in Section 1.2 is characterized by a gap in time and a drop in NIC from a value above 7 to 0 on either or both sides of the gap, which highlights the need to maintain the temporal features of the data. This problem is modeled as a supervised binary classification problem with two primary features: Time, transformed into time differences (Δt) to highlight gaps, and NIC, treated as a categorical variable, with a cardinality of 12 (discrete numbers ranging from 0 to 11).

The sliding window technique is a key part of the data modeling process, and it plays a significant role in preserving the temporal characteristics of the data. Essentially, this technique involves moving a window of a fixed size over the data, allowing us to capture the temporal features. As shown in Figure 9, the green box represents a positive sample, where a gap in time and a drop in NIC are observed on both sides of the gap, while the blue boxes depict negative samples. The problem is represented mathematically in Equation (1), where $N \in \{0, 1, \ldots, 11\}$ and $\hat{y} \in \{0, 1\}$. The function f represents the ML model that takes as input a flattened window of difference in time and NIC.

$$f(\Delta t_n, N_n, \Delta t_{n+1}, N_{n+1} \ldots \Delta t_{n+29}, N_{n+29}) = \hat{y} \tag{1}$$

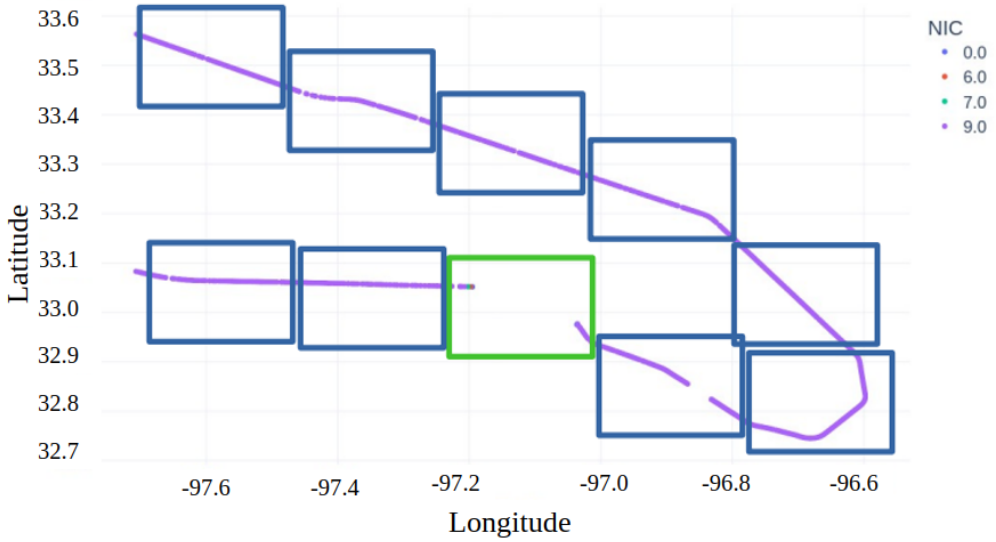

Figure 9. Data modeling: sliding window technique with a positive sample (green box) and negative samples (blue boxes).

To overcome labeling challenges, a synthetic data generator was developed based on real-world ADS-B data properties. The essence of the generator is represented as pseudocode in Algorithm 2, while the code can be found at the DECS Lab's GitHub repository y [40]. Our dataset comprised of three sets: the training set contained 126,000 synthetic data samples, the test set with 54,000 synthetic data points, and the validation set contained

600 real-world positive samples. The validation set's composition reflects the emphasis on recall for rare event detection. The challenge at hand involves detecting rare events in high-dimensional data with temporal characteristics that render traditional tests for normality unfeasible. This requirement, in addition to the critical nature of the application, led us to favor well-tested and renowned implementations of ML algorithms whose properties are well established over implementing new models from scratch, ensuring reliability and efficiency in safety-critical and big data contexts.

Algorithm 2 Train and Test data generation algorithm

```
Initialize empty lists targets, time_diffs, and nics
for i in range (10,000) do
    Initialize nic_values and time_diff with defaults
    Generate a gap with random duration and location
    if gap exists then
        Determine if there is an NIC drop
        if NIC drop exists then
            Choose drop type
            Update NIC values accordingly
        end if
        Append time_diff and nic_values to lists
    else
        Append default time_diff and nic_values
    end if
    Append target label (1 for NIC drop, 0 for no drop)
end for
for i in range (10,000) do
    Initialize nic_values and time_diff with defaults
    Generate a gap with random duration and location
    if gap exists then
        Append time_diff and nic_values to lists
    else
        Append default time_diff and nic_values
    end if
    Append target label (0 for no NIC drop)
end for
```

A diverse set of conventional ML algorithms are applied from Python's Scikit Learn Library. Models from different categories including linear, Bayesian, stochastic, distance-based, tree-based, and network-based. Logistic regression offers simplicity and interpretability but struggles with non-linear relationships. Bayesian models, like Naive Bayes, perform well with limited data and high dimensionality but make strong assumptions about feature independence. Support Vector Machines (SVM) are known to perform well on data with higher dimensions but are computationally expensive. Stochastic models excel with large datasets but may take time to converge. Random forest handles non-linearity really well but tends to lose interpretability in complex scenarios. Neural networks capture intricate relationships in the data but require large volumes of data and are computationally expensive to train and run. Initially, they are trained and tested on 180,000 simulated data points with a 70–30 split; the models with a high False Positive Rate (FPR) and False Negative Rate (FNR) are ignored, preferring models with a high TPR. In the next step, the remaining models are tested for prediction variability by simulating 5 more iterations of data from the data generator, and the models will be tested on prediction variability on the test set. Models with a low prediction variance would be studied further using permutation feature importance to understand which features the model prioritized by the different ML models to make its prediction [41]. Permutation feature importance measures how much the model's performance metric decreases when a feature's values are randomly shuffled,

thereby breaking the relationship between the feature and the target mathematically as shown in Equation (2).

$$\text{importance}(X_i) = \frac{1}{n_{\text{repeats}}} \sum_{r=1}^{n_{\text{repeats}}} \left(\text{score} - \text{score}_{\text{perm}(X_i)} \right) \tag{2}$$

where

- Importance(X_i) is the importance of feature X_i;
- n_{repeats} is the number of repetitions;
- Score is the model score on the original dataset;
- $\text{Score}_{\text{perm}(X_i)}$ is the model score after permuting feature X_i.

The models are then validated on the 600 real-world positive samples. Models with high TPR and low FNR will be preferred. Finally, given that the models are intended for real-time predictions in big-data scenarios, the models will tested by iteratively increasing the sample size in multiples of two, from 1 to 2048. These steps not only give insight into model prediction capabilities on simulated and real-world data but also explain the prioritization of features, which helps explain how models interpret data to make predictions. By following these steps, a computationally efficient and accurate model will be selected.

The rarity of the positive samples is a primary factor in model performance assessment. Traditional metrics like accuracy are misleading in such scenarios. To address this, the confusion matrix metrics, including the true positive rate (TPR), the false positive rate (FPR), and the positive predictive value (PP), will be employed. Focusing on these metrics over accuracy ensures the predominance of negative samples does not skew the assessment.

3. Results and Discussion

The methodology and the results are detailed in this section, while an overview is visualized in Figure 10.

1. **Model Training**: The confusion matrix, shown as a bar chart in Figure 11, depicts the result of training the model on 180,000 simulated data points with 70–30 train–test split. Ideal models have a high TPR, and PP and a low FPR and FNR. In contrast, Naive Bayes exhibited a high FPR, while SGD classifier, SVM, and logistic regression have a high FNR.
2. **Variance Testing and Model Explainability**: The models filtered from the previous step were used to test for prediction variance. This step is performed to ensure that the performance of the models is not due to a statistical fluke in the generation process. NN, random forest, and MLP classifier were tested by iterating five runs of newly created 180,000 data points each time and performing a 70–30 train–test split. All of the models exhibited a low variance, indicating reliable learning patterns that do not arise from statistical anomalies. The results of running 150 iterations of permutation feature importance on these four models revealed that the models prioritized NIC similarly but varied slightly on how it was prioritized. Figures 12 and 13 visualize these findings in a line chart with the time step of ΔT and NIC on the X-axis and feature importance on the Y-axis. We found that MLP and dense neural networks prioritized the initial and final time steps of a window the most, random forest and decision tree classifiers placed them at a relatively lower importance. The central part of a window of 30 s seems to be the most important. MLP prioritizes this part of the window the least in relation to random forest, decision tree classifier, and dense neural network.
3. **Real-world data performance**: The models were evaluated on 600 real-world positive samples, and the results are outlined in Table 4. The decision tree classifier performed poorly on real-world data, with a TPR of 23% and an FNR of 76.4%, leading to its exclusion from further steps. In correlation with the model explainability, MLP, and

dense neural network prioritized data points at the beginning and end of each window more than random forest and decision tree, causing them to capture all instances of drops, even if they occur at the beginning or end of a window, which may have been missed by random forest and decision tree classifier, resulting in a lower TRP compared to dense neural network and MLP.

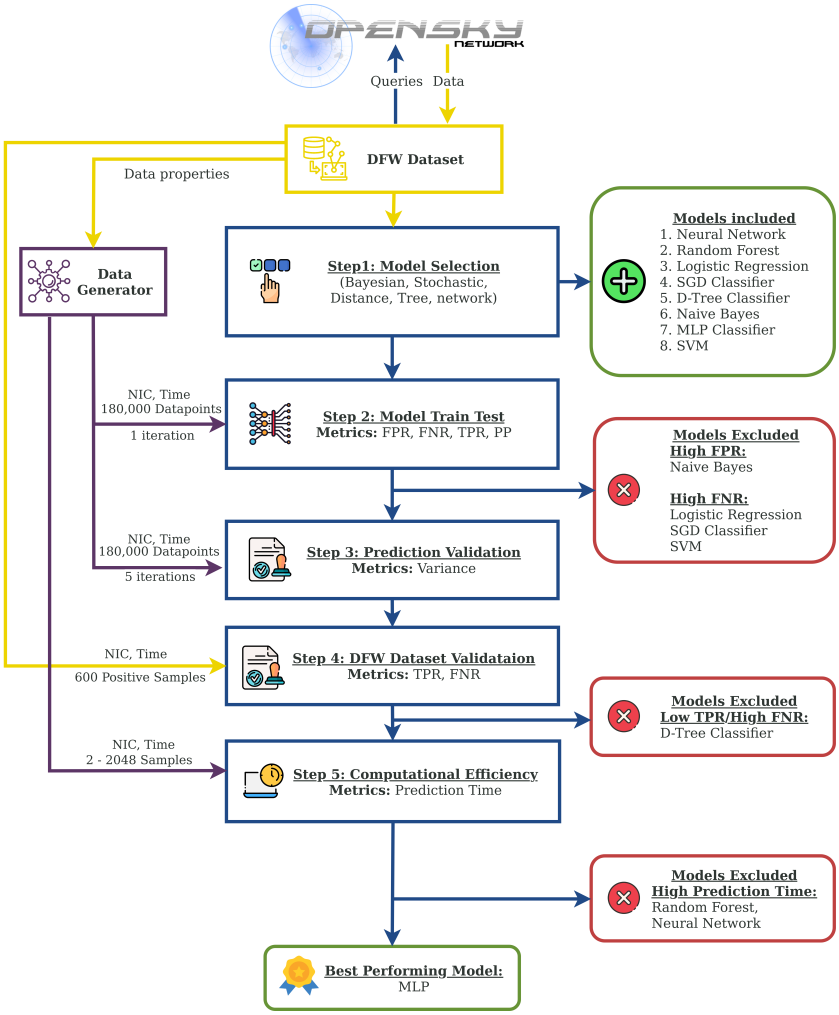

Figure 10. Model training process and results.

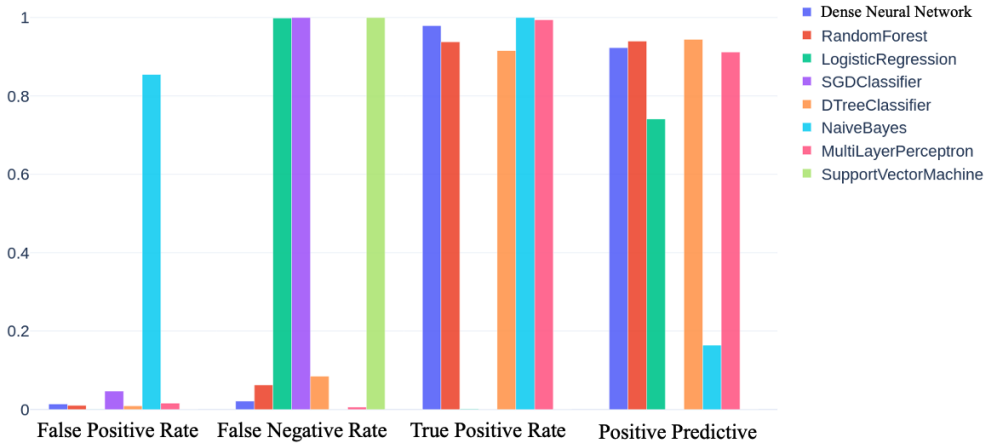

Figure 11. Model performance on test data, 20% of generated training data.

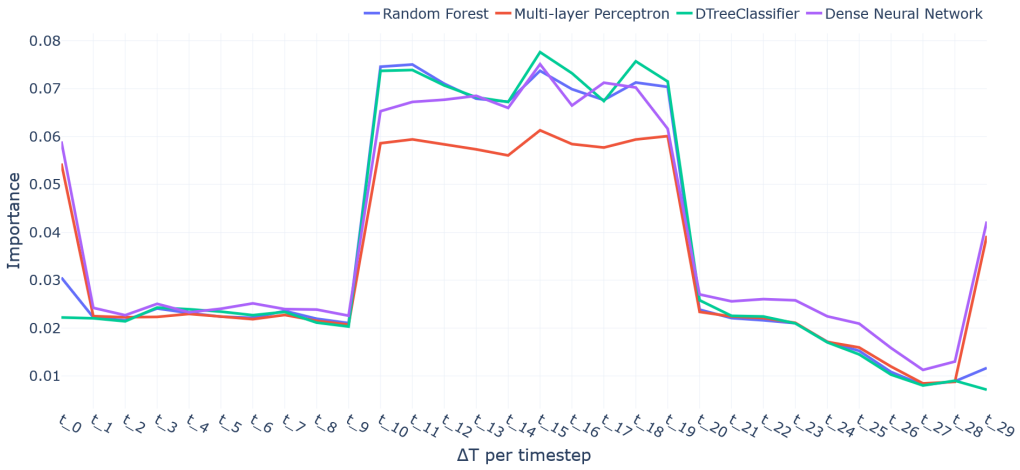

Figure 12. Permutation feature importance for ΔT.

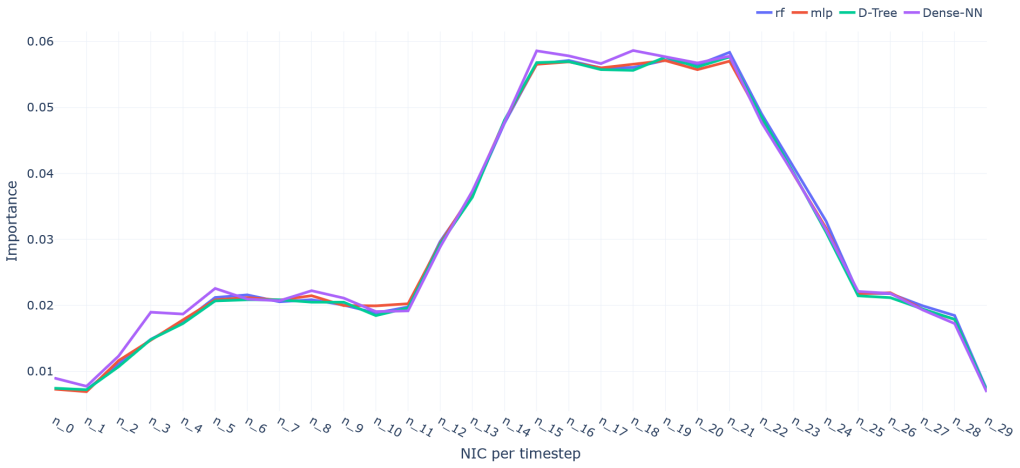

Figure 13. Permutation feature importance for NIC.

Table 4. Performance of ML models on real-world 600 positive samples. Best performing model is highlighted in bold.

Model	True Positive Rate	False Negative Rate
RF	99.536	0.464
Decision Tree	23.648	76.352
MLP	**99.845**	**0.154**
Dense NN	99.845	0.155

4. **Computational Efficiency**: The models selected from the previous step (MLP, random forest, and dense NN) were tested for their computational efficiency by iteratively increasing the input sample size from 1 to 2048, in multiples of two. The results, visualized in Figure 14, show that for dense NN and random forest, the prediction time grows as the number of samples increases, while the prediction time remains mostly unaltered for MLP. Thus, MLP is the best-performing model in terms of prediction accuracy and time, reducing prediction time by up to 10× and 16× compared to random forest, which showed a 5× reduction in computation time for 2000 samples.

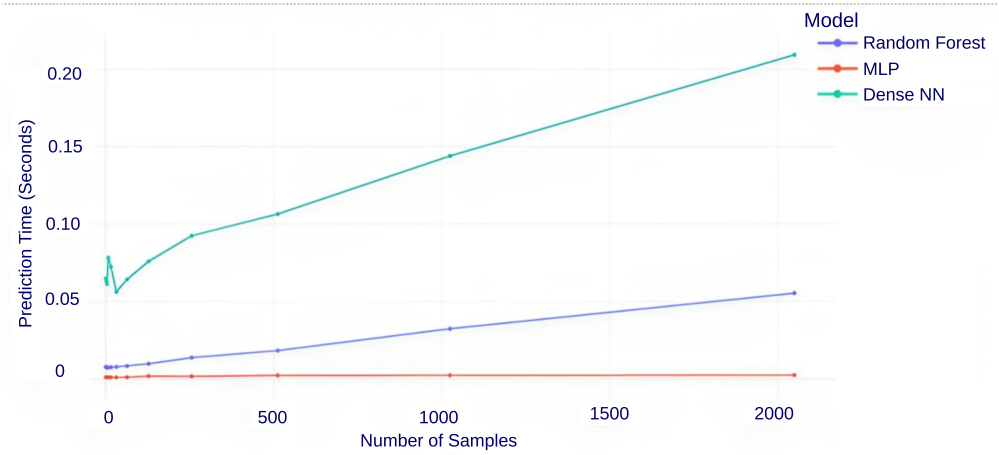

Figure 14. Prediction time (seconds) by number of samples.

4. Conclusions

The literature review revealed a consistent pattern of GPS interference in ADS-B data. Multiple studies agree that GPS interference typically manifests as a temporal gap in ADS-B messages, accompanied by a decrease in Navigation Integrity Category (NIC) values either before, after, or on both sides of this gap. This characteristic pattern serves as the primary indicator of GPS interference events in ADS-B data streams. The investigation into GPS interference using ADS-B data revealed significant fluctuations in NIC during the event, and 600 aircraft were identified that contained the GPS interference pattern. A notable finding was that aircraft as high as 9 to 15 thousand meters experienced a drop in NIC. In terms of direction, aircraft heading south and southeast tend to have a higher drop in NIC compared to other directions. The most typical change in NIC was found to be from 9 to 0 and back to 9. In detecting the GPS interference pattern, three algorithms performed the best: random forest, dense NN, and MLP. This is attributed to the fact that these models prioritize points that are at the beginning and end of a window and a mild correlation between the extent of prioritization and TPR. The models were able to achieve a high TPR of up to 99.845 and a very low FNR of up to 0.154. This allows predictions with low Type-II error, which indicates that it is very unlikely to misclassify an aircraft that exhibited the GPS interference pattern. Though random forest, dense NN, and multi-layer perceptron algorithms performed well in terms of TPR, the prediction times of these algorithms are significantly different. Iteratively increasing the number of samples from 2 to 2048 in multiples of two, it was found that dense NN's prediction time grew exponentially with the number of samples. In terms of prediction speed, it was found that MLP's prediction speed remained the lowest, even when the number of samples grew. MLP is able to predict accurately and efficiently. This finding lays the groundwork for future applications that can leverage the capabilities of MLP to implement real-time ML-based monitoring systems that can efficiently detect GPS interference in ADS-B data, contributing to aviation safety and air space management.

The limitations of this work are as follows:

1. The current implementation of the Python script starts with a large download from the state vectors table. Although this is robust to server timeouts, it would have to be improved to implement the same methodology applied to the operational status table.
2. The ML model has been trained on synthetic data based on a GPS interference pattern described in the literature. For a more accurate detection, actual pilot accounts of GPS interference must be used.
3. The current methodology only considers front door interference (interference caused by external sources). However, backdoor interference (caused by internal sources) is not accounted for and may show up as false positives.

Further research is warranted to address these limitations. This is described in more detail in the following subsection.

Future Work

The current implementation of the OpenSky downloader will be extended to overcome limitations by implementing the same download methodology used in the operational status download. As follow-up work to this paper, a larger dataset of manually labeled instances of the GPS interference pattern will be created. ML models will be trained to detect such patterns in live ADS-B data. This will enable the creation of an interface that displays areas of GPS interference in near-real time to serve as a guideline to the ATC and pilots alike. A series of non-ML algorithms will be developed that are able to detect the GPS interference pattern, which will be compared to MLP (the best-performing model identified in this work) to make a decision between ML and non-ML algorithms for GPS interference detection.

Author Contributions: A.R.R., A.S., P.R. and W.S. conceived the experiment(s), A.R.R. and A.S. conducted the experiment, A.R.R. and A.S. analyzed the results. All authors have read and agreed to the published version of the manuscript.

Funding: The FAA has sponsored this project through the Center of Excellence for Unmanned Aircraft Systems. However, the agency neither endorses nor rejects the findings of this research. The presentation of this information is in the interest of invoking technical community comment on the results and conclusions of the research. Award No: 15-C-UAS-UND-030. Award Name: Mitigating GPS and ADS-B Risks for UAS. Program: ASSURE UAS COE

Data Availability Statement: Restrictions apply to the availability of these data. Data were obtained from the Opensky Network and is available for research purposes at https://opensky-network.org/ by applying for an educational account.

Acknowledgments: The authors thank the OpenSky Network for their invaluable contribution to this research. Their provision of comprehensive and accurate ADS-B data was instrumental in completing this work. The data obtained from the OpenSky Network enabled insightful analysis and deepened the understanding of the subject matter, significantly enhancing the quality and relevance of this study. We sincerely appreciate their support and the opportunity to utilize such a rich dataset. Their commitment to facilitating academic research and fostering knowledge in the field is commendable. The graphs in this paper were created using Plotly, an open-source graphing library that helps users create interactive plots. This manuscript is based on work previously published in the Akshay Ram Ramchandra's Master's thesis titled 'Anomaly Detection And Trajectory Prediction Models For ADS-B Datasets', which was submitted to the University of North Dakota. Sections of this manuscript have been adapted from the thesis with permission

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GPS	Global Positioning Systems
DFW	Dallas Fort Worth
NIC	Navigation Integrity Category
NN	Neural Network
MLP	Multi-Layer Perceptron
TPR	True Positive Rate
VHF	Very High Frequency
FAA	Federal Aviation Authority
NextGen	Next Generation
ATC	Air Traffic Control
NAC	Navigation Accuracy Category
SIL	Surveillance Integrity Level
HDFS	Hadoop Distributed File System

AI	Artificial Intelligence
FPR	False Positive Rate
FNR	False Negative Rate
PPV	Positive Predictive Value
IQR	Interquartile Range
LEO	Low Earth Orbit
PRN	pseudo-random number
SSH	Secure Shell
SQL	Structured Query Language
CSV	Comma Separated Value

References

1. Harris, M. Military Tests that Jam and Spoof GPS Signals are an Accident Waiting to Happen. *IEEE Spectr.* **2021**, *58*, 22–27. [CrossRef]
2. McCallie, D.L. *Exploring Potential ADS-B Vulnerabilites in the FAA's Nextgen Air Transportation System*; Technical Report; Air Force Institute of Technology: Wright-Patterson AFB, OH, USA, 2011.
3. Huang, G.; Taylor, B.; Akopian, D. A Low-Cost Approach of Magnetic Field-Based Location Validation for Global Navigation Satellite Systems. *IEEE Trans. Instrum. Meas.* **2019**, *68*, 4937–4944. [CrossRef]
4. Nadarajah, N.; Teunissen, P.J.; Raziq, N. Instantaneous GPS–Galileo attitude determination: Single-frequency performance in satellite-deprived environments. *IEEE Trans. Veh. Technol.* **2013**, *62*, 2963–2976. [CrossRef]
5. Space-Based Positional Navigation and Timing Twenty-Fourth Meeting Minutes. National Advisory Board. 2019. Available online: https://www.gps.gov/governance/advisory/meetings/2019-11/minutes.pdf (accessed on 11 November 2023).
6. CISA. Global Positioning System (GPS) Interference. CISA Insights. 2022. Available online: https://www.cisa.gov/sites/default/files/publications/CISA-Insights_GPS-Interference_508.pdf (accessed on 11 November 2023).
7. Warwick, G. GPS Interference Grows as a Concern for Civil Aviation. 2024. Available online: https://aviationweek.com/aerospace/connected-aerospace/gps-interference-grows-concern-civil-aviation (accessed on 20 July 2024)
8. Goward, D. What Happened to GPS in Denver? Available online: https://www.gpsworld.com/what-happened-to-gps-in-denver/ (accessed on 15 November 2023).
9. Goodin, D. GPS Interference Caused the FAA to Reroute Texas Air Traffic. Experts Stumped. Available online: https://arstechnica.com/information-technology/2022/10/cause-is-unknown-for-mysterious-gps-outage-that-rerouted-texas-air-traffic/ (accessed on 15 November 2023).
10. Jie, H.; Zhao, Z.; Zeng, Y.; Chang, Y.; Fan, F.; Wang, C.; See, K.Y. A review of intentional electromagnetic interference in power electronics: Conducted and radiated susceptibility. *IET Power Electron.* **2024**. [CrossRef]
11. Hassan, T.J.; Jangula, J.; Ramchandra, A.R.; Sugunaraj, N.; Chandar, B.S.; Rajagopalan, P.; Rahman, F.; Ranganathan, P.; Adams, R. UAS-Guided Analysis of Electric and Magnetic Field Distribution in High-Voltage Transmission Lines (Tx) and Multi-Stage Hybrid Machine Learning Models for Battery Drain Estimation. *IEEE Access* **2024**, *12*, 4911–4939. [CrossRef]
12. New FAA Files Reveal Private Aircrafts Affected by U.S. Military's GPS Jamming During Tests. 2024. Available online: https://www.engineering.com/new-faa-files-reveal-private-aircrafts-affected-by-u-s-militarys-gps-jamming-during-tests/ (accessed on 15 November 2023).
13. RTCA Special Committee. *Minimum Aviation System Performance Standards for Automatic Dependent Surveillance Broadcast (ADS-B)*. Technical Report; RTCA Special Committee: Washington, DC, USA, 1998.
14. Wu, Z.; Shang, T.; Guo, A. Security issues in automatic dependent surveillance-broadcast (ads-B): A survey. *IEEE Access* **2020**, *8*, 122147–122167. [CrossRef]
15. Syd Ali, B.; Schuster, W.; Ochieng, W.; Majumdar, A. Analysis of anomalies in ADS-B and its GPS data. *GPS Solut.* **2016**, *20*, 429–438. [CrossRef]
16. Sun, J. *The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals*, 2nd ed.; TU Delft OPEN Publishing: Delft, The Netherlands, 2021. [CrossRef]
17. Public ADS-B Performance Report User's Guide. Available online: https://adsbperformance.faa.gov/PAPRUsersGuide.pdf (accessed on 15 November 2023).
18. Schäfer, M.; Strohmeier, M.; Lenders, V.; Martinovic, I.; Wilhelm, M. Bringing up OpenSky: A large-scale ADS-B sensor network for research. In Proceedings of the IPSN-14, 13th International Symposium on Information Processing in Sensor Networks, Berlin, Germany, 15–17 April 2014; pp. 83–94.
19. Pothana, P.; Joy, J.; Snyder, P.; Vidhyadharan, S. UAS Air-Risk Assessment In and Around Airports. In Proceedings of the 2023 Integrated Communication, Navigation and Surveillance Conference (ICNS), Herndon, VA, USA, 18–20 April 2023; pp. 1–11.
20. Ullrich, M.; Pothana, P.; Thornby, J.; Snyder, P.; Vidhyadharan, S. Determining the Saturation Point for UAV Operations in Airport Environments: A Probabilistic Approach. In Proceedings of the 2024 Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon, VA, USA, 23–25 April 2024; pp. 1–11.
21. Gallego-Garcia, P.; Hong, S.L.; Bollen, N.; Dellicour, S.; Baele, G.; Suchard, M.A.; Lemey, P.; Posada, D. Dispersal history of SARS-CoV-2 variants Alpha, Delta, and Omicron (BA.1) in Spain. *medRxiv* **2024**. [CrossRef]

22. Olive, X.; Strohmeier, M.; Sun, J.; Tresoldi, G. OpenSky Report 2024: Analysis of Global Flight Contrail Formation and Mitigation Potential. In Proceedings of the 2024 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC), San Diego, CA, USA, 29 September–3 October 2024.
23. OpenSky Network. Historical Flight Data Documentation. 2023. Available online: https://opensky-network.org/data/historical-flight-data (accessed on 28 September 2023).
24. Apache Impala. 2023. Available online: https://impala.apache.org/ (accessed on 2 August 2023).
25. Murrian, M.J.; Narula, L.; Iannucci, P.A.; Budzien, S.; O'Hanlon, B.W.; Psiaki, M.L.; Humphreys, T.E. First results from three years of GNSS interference monitoring from low Earth orbit. *Navigation* **2021**, *68*, 673–685. [CrossRef]
26. Abraha, K.E.; Frisk, A.; Wiklund, P. GNSS interference monitoring and detection based on the Swedish CORS network SWEPOS. *J. Geod. Sci.* **2024**, *14*, 20220157. [CrossRef]
27. O'Mahony, G.D.; McCarthy, K.G.; Harris, P.J.; Murphy, C.C. Developing novel low complexity models using received in-phase and quadrature-phase samples for interference detection and classification in Wireless Sensor Network and GPS edge devices. *Ad Hoc Netw.* **2021**, *120*, 102562. [CrossRef]
28. Sun, K.; Zhang, T. A new GNSS interference detection method based on rearranged wavelet–hough transform. *Sensors* **2021**, *21*, 1714. [CrossRef] [PubMed]
29. Patil, A.; Phelts, R.E.; Chen, Y.H.; Lo, S.; Walter, T. Detecting Space Based Interference on GNSS Signals. In Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023), Denver, CO, USA, 11–15 September 2023; pp. 1232–1244.
30. Lukeš, P.; Topková, T.; Vlček, T.; Pleninger, S. Recognition of GNSS Jamming Patterns in ADS-B Data. In Proceedings of the 2020 New Trends in Civil Aviation (NTCA), Prague, Czech Republic, 23–24 November 2020; pp. 9–15.
31. Liu, Z.; Lo, S.; Walter, T. Characterization of ADS-B performance under GNSS interference. In Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020), Online, 21–25 September 2020; pp. 3581–3591.
32. Ala'Darabseh, E.B.; Tedongmo, B. Detecting gps jamming incidents in opensky data. In Proceedings of the 7th OpenSky Workshop, Zurich, Switzerland, 21–22 November 2019; Volume 67, pp. 97–108.
33. Jonáš, P.; Vitan, V. Detection and localization of GNSS radio interference using ADS-B data. In Proceedings of the 2019 International Conference on Military Technologies (ICMT), Brno, Czech Republic, 30–31 May 2019; pp. 1–5.
34. Liu, Z.; Lo, S.; Walter, T.; Blanch, J. Real-time detection and localization of GNSS interference source. In Proceedings of the 35th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2022), Denver, CO, USA, 19–23 September 2022; pp. 3731–3742.
35. Balaei, A.T.; Dempster, A.G. A statistical inference technique for GPS interference detection. *IEEE Trans. Aerosp. Electron. Syst.* **2009**, *45*, 1499–1511. [CrossRef]
36. Liu, Z.; Lo, S.; Walter, T. GNSS Interference Detection Using Machine Learning Algorithms on ADS-B Data. In Proceedings of the 34th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2021), Online, 20–24 September 2021; pp. 4305–4315.
37. Ramchandra, A.R. Open_SKY: Query the OpenSky Network Database. Commit e7a50bd. 2022. Available online: https://github.com/DECSResearch/open_sky (accessed on 1 July 2024).
38. EU Agency for the Space Programme. What Is SBAS? 2024. Available online: https://www.euspa.europa.eu/eu-space-programme/egnos/what-sbas (accessed on 22 July 2024).
39. European Space Agency. SBAS Fundamentals. 2024. Available online: https://gssc.esa.int/navipedia/index.php/SBAS_Fundamentals(accessed on 22 July 2024).
40. Ramchandra, A.R. GPS Interference Detection. 2023. Available online: https://github.com/DECSResearch/Interference-Detection (accessed on 12 December 2023).
41. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-Learn: Permutation Feature Importance. Version 0.24.2. 2011. Available online: https://scikit-learn.org/stable/modules/permutation_importance.html (accessed on 20 July 2024).