

# Anomaly Detection in ADS-B Air Traffic Data Using Distributed Machine Learning <sup>†</sup>

Maria Babanova <sup>1</sup>, Radoslav Furnadzhiev <sup>1,2,\*</sup> and Mitko Shopov <sup>1,2</sup>

<sup>1</sup> Department of Computer Systems and Technologies, Faculty of Electronics and Automation, Technical University of Sofia Branch Plovdiv, 25 Ts. Diustabanov St., 4000 Plovdiv, Bulgaria; m.babanova@tu-plovdiv.bg (M.B.); mshopov@tu-plovdiv.bg (M.S.)

<sup>2</sup> Center of Competence “Smart Mechatronic, Eco- and Energy-Saving Systems and Technologies”, 4000 Plovdiv, Bulgaria

\* Correspondence: rfurnadzhiev@tu-plovdiv.bg

<sup>†</sup> Presented at the 15th International Scientific Conference TechSys 2026—Engineering, Technologies and Systems, Plovdiv, Bulgaria, 14–16 May 2026.

## Abstract

This paper presents a comparative study of two unsupervised machine learning methods, Isolation Forest (IF) and Gaussian Mixture Model (GMM), for anomaly detection in Automatic Dependent Surveillance–Broadcast (ADS-B) data. While ADS-B provides critical flight telemetry, its high volume and inherent noise pose significant challenges for automated surveillance. To address this, we developed a distributed Extract Transform Load (ETL) pipeline using Apache Spark to analyze a dataset of 4.82 billion records. The results show a low Pearson correlation between the two models, suggesting that they capture complementary types of anomalies: IF identifies structural outliers through recursive partitioning, while GMM detects statistical deviations from learned probability densities. This study demonstrates that an ensemble of lightweight methods can effectively scale to billion-record datasets, with IF offering superior computational efficiency (~1.7 h) compared to GMM (~4.8 h). These results provide a robust baseline for developing scalable, real-time aviation monitoring systems to enhance global air traffic safety.

**Keywords:** Automatic Dependent Surveillance–Broadcast; anomaly detection; machine learning; unsupervised learning; Isolation Forest; Gaussian Mixture Model

## 1. Introduction

Anomalies in data often indicate potential problems, risks, or new opportunities. The development of systems for analyzing and detecting anomalies in air traffic data aims to support the timely identification of unusual situations, reduce the likelihood of human errors, and optimize the analysis of large volumes of data. In the aviation context, where safety is critical, identifying such deviations is essential to preventing incidents and improving air traffic management. Automatic Dependent Surveillance–Broadcast (ADS-B) is a modern surveillance technology that enables aircraft to automatically broadcast information about their identification, position, speed, and status to other aircraft and nearby ground stations [1]. These messages provide a rich source of data for analyzing flight behavior and detecting unusual patterns. Anomalies are generally defined as patterns in the data that deviate from expected normal behavior [2,3]. In aviation data analysis, anomaly detection is often used to identify unusual operational conditions, safety-related events, or irregular aircraft trajectories that may indicate potential risks or inefficiencies in air traffic



Academic Editors: Nikola G. Shakev,  
Valyo N. Nikolov, Nikolay R.  
Kakanakov and Sevil A.  
Ahmed-Shieva

Published: 20 July 2026

**Copyright:** © 2026 by the authors.  
Licensee MDPI, Basel, Switzerland.  
This article is an open access article  
distributed under the terms and  
conditions of the [Creative Commons  
Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

operations [4]. The analysis of ADS-B data presents several challenges, including the large volume of data, its dynamic nature, and the need for efficient processing close to real time. Consequently, machine learning methods are increasingly used for automated anomaly detection and for identifying unusual behavioral patterns in air traffic. The objective of this study is to compare two machine learning algorithms, Isolation Forest and Gaussian Mixture Model, for anomaly detection in datasets of ADS-B messages, using an ETL-based data processing pipeline. Isolation Forest is an efficient anomaly detection algorithm that isolates anomalous observations by recursively partitioning the data space [5]. Gaussian Mixture Models are probabilistic models that represent data distributions as a combination of multiple Gaussian components and have been successfully applied to identify abnormal patterns in aviation flight data [6]. The analysis aims to evaluate the applicability of these algorithms to air traffic data processing and to support the development of more efficient monitoring and analytical systems.

## 2. Background and Related Work

Automatic Dependent Surveillance–Broadcast (ADS-B) is a key component of modern air traffic surveillance because it enables aircraft to broadcast position, velocity, altitude, and identification data at frequent intervals. Its value for monitoring and anomaly detection, however, depends not only on message availability but also on data quality, consistency, and credibility. It has been shown that ADS-B data evaluation must account for synchronization, accuracy, and reliability issues, as anomalous observations may reflect true operational deviations, sensor imperfections, or integrity problems rather than genuinely abnormal flight behavior [1]. The broader aviation literature has established anomaly detection as an important use of machine learning for flight-data analysis. Previous research on machine-learning techniques for flight-data analysis has highlighted the challenges of working with high-dimensional, time-dependent, and only partially labeled aviation datasets [3]. More recent research has proposed machine-learning models for online anomaly detection in flight operations, emphasizing the practical need for methods that can operate in streaming or near-real-time environments while maintaining acceptable computational cost [3]. This makes unsupervised and semi-supervised approaches particularly relevant in aviation settings, where anomalous events are rare and labeled data are limited. Among unsupervised methods, Isolation Forest remains one of the most established baselines because it isolates anomalies through recursive random partitioning rather than by explicitly estimating the distribution of normal data. Anomalous instances tend to be isolated and occur in fewer partitions, making this method both efficient and appropriate for large datasets [5]. This is attractive for ADS-B analysis, where surveillance streams may contain large volumes of records and heterogeneous anomaly types, including isolated outliers and subtler deviations from routine operational patterns [7]. A complementary probabilistic method is the Gaussian Mixture Model (GMM), which represents normal behavior as a mixture of Gaussian components. GMM-based anomaly detection can support flight operations and safety monitoring by identifying flights with unusual data patterns relative to learned normal behavior [6]. Compared with tree-based approaches, GMMs provide a probabilistic interpretation of abnormality and can model multimodal patterns, which is useful in aviation because normal behavior varies across aircraft types, routes, phases of flight, and operating conditions. For methodological context, it is also useful to consider comparative work outside aviation. A comparison of 13 traditional and deep learning-based anomaly-detection methods on streaming datasets found that deep models often perform better when the data exhibit temporal structure, such as trend, seasonality, and change points [8]. Although their study is not aviation-specific, it is relevant because ADS-B data are also sequential, non-stationary, and streaming. This reference, therefore,

strengthens the justification for comparing conceptually distinct anomaly-detection approaches in an ADS-B setting. Taken together, the literature supports three points. First, ADS-B anomaly detection must be grounded in data quality and surveillance reliability considerations [9]. Second, machine-learning methods are increasingly used for flight and surveillance anomaly detection because they can handle complex, weakly labeled data [3]. Third, Isolation Forest and GMM provide two complementary unsupervised perspectives on abnormality: one isolates rare observations through recursive partitioning, while the other evaluates them relative to an estimated probabilistic model of normal behavior [5,6]. At the same time, there is still limited comparative evidence on lightweight unsupervised methods applied specifically to noisy ADS-B streams, which motivates the present study.

### 3. Anomaly Detection Algorithms

#### 3.1. Isolation Forest

Isolation Forest is an ensemble anomaly detection algorithm that exploits the property that anomalous observations, being few and structurally atypical, require fewer binary partitions to isolate in a random decision tree than normal observations [5]. The algorithm constructs a forest of  $n\_estimators$  randomized Isolation Trees by recursively splitting the feature space at uniformly sampled feature-threshold pairs. For an observation  $x$ , the path length  $h(x)$  in a single tree minus the number of splits required to isolate it serves as a proxy for anomalousness. The average path length across the trees is transformed into an anomaly score, where shorter paths indicate more anomalous observations. This makes Isolation Forest efficient and suitable for high-dimensional, large-scale datasets such as ADS-B data. The normalized anomaly score is:

$$s(x, n) = 2^{\frac{-E(h(x))}{c(n)}} \quad (1)$$

where  $E[h(x)]$  is the expected path length over the forest,  $c(n)$  is the expected path length for a dataset of size  $n$  used for normalization, and  $c(n) \approx 2H(n-1) - \frac{2(n-1)}{n}$  with  $H(i) \approx \ln(i) + 0.5772$  (Euler–Mascheroni constant). Scores approaching 1 indicate anomalies; scores below 0.5 indicate normal observations.

The algorithm is implemented using the scikit-learn library [10] via Spark's mapIn-Pandas API. The scaled features dataset is repartitioned into  $[N/batch\_size]$  batches of 100,000 records each, and an independent scikit-learn IsolationForest instance is trained and scored on each batch within the executor. The partition scheme is random rather than by *flight\_id* so that each batch contains a representative cross-section of flight types, providing a population reference for anomaly boundary estimation.

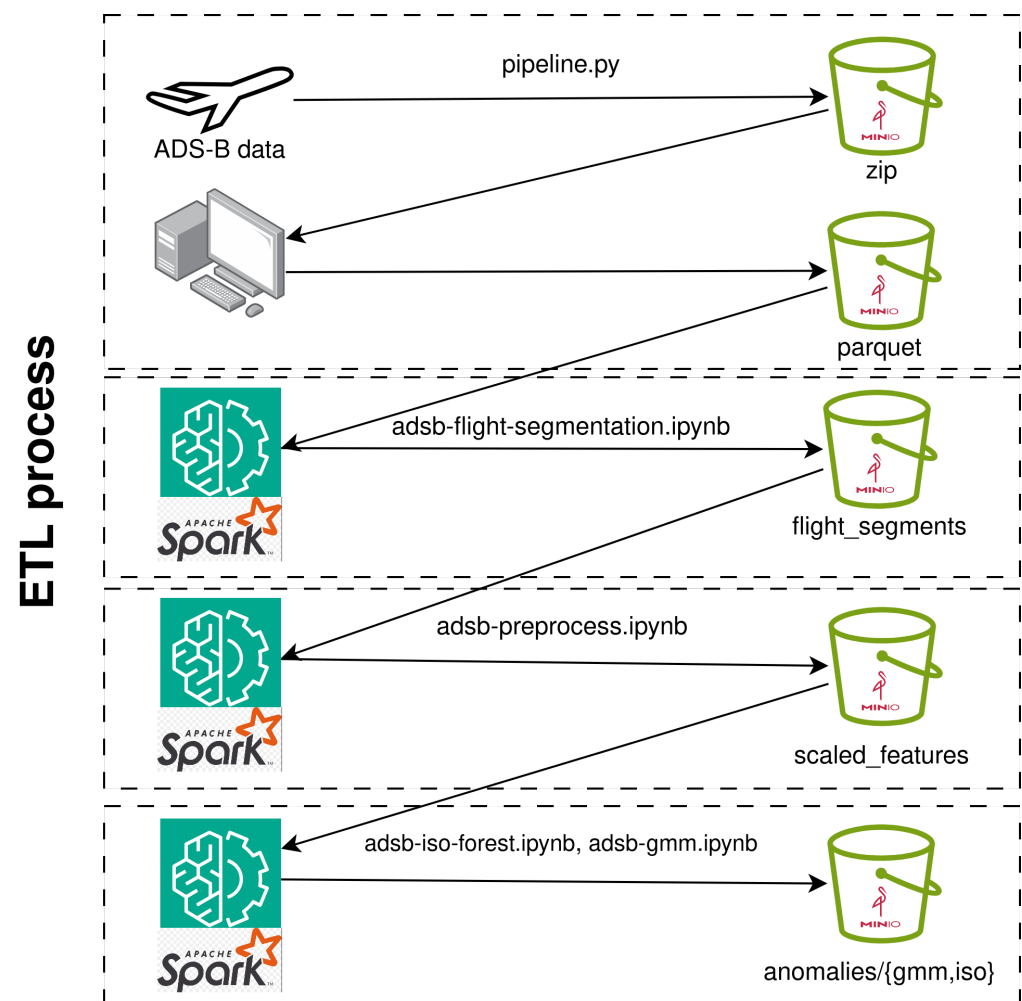
#### 3.2. Gaussian Mixture Model (GMM)

The Gaussian Mixture Model (GMM) represents the data distribution as a weighted sum of  $K$  multivariate Gaussian components, which are fitted using the Expectation–Maximization (EM) algorithm. The EM algorithm iterates between an E-step, which computes posterior component membership probabilities for each observation, and an M-step, which updates component weights  $\pi_k$ , means  $\mu_k$ , and covariance matrices  $\Sigma_k$  to maximize the expected log-likelihood. Anomaly detection is performed by evaluating the log of the maximum component probability for each record; observations below a percentile threshold of the resulting score distribution are classified as anomalous. A percentile-based threshold produces a consistent anomaly rate independently of dataset size and the absolute scale of log-probability values. Training the full EM algorithm on 4.82 billion records is computationally prohibitive due to the per-iteration full-dataset pass required. The model is trained on a randomly sampled 10% subset (482 million rows), and the subset is persisted

to disk before training to prevent Spark from re-evaluating the sampling lineage on each EM iteration. After convergence, the fitted model is applied to the complete dataset in a single scoring pass using Spark MLlib's distributed `GaussianMixture.transform()`.

#### 4. System Architecture and ETL Pipeline

The system is built as a modular ETL pipeline. Each stage is implemented as a self-contained Python 3.12 script or Jupyter Notebook 7.6, allowing independent development and scaling. The pipeline spans four sequential stages, each implemented as a self-contained Python script or Jupyter Notebook that consumes the output of the previous stage and produces a new materialized dataset. All computations are performed on an Apache Spark 3.5.0 [11] standalone cluster with 32 CPU cores and 96 GB of RAM. Figure 1 summarizes the stages and their outputs.



**Figure 1.** Pipeline stage summary.

##### 4.1. Stage 1—Extract

Raw ADS-B position data is sourced from the `adsblol globe_history` archive, a community-maintained repository published as daily GitHub releases. Each release consists of alphabetically indexed split tar archives (`.tar.aa`, `.tar.ab`, ...), each containing compressed binary JSON trace files in the `readsb` format, one file per aircraft. The volunteer ground-receiver network providing this data offers dense coverage over Europe, North America, and East Asia; coverage over oceanic and polar regions is sparse or absent, producing systematic gaps in the message stream for aircraft transiting those areas. The first stage of the process, implemented in `pipeline.py`, iterates over a specified date range, constructing

the release tag for each day and downloading split parts sequentially until a 404 response indicates no further parts exist. Existing files are not re-downloaded, making the stage idempotent. Each trace file is unzipped and decoded. The decoded JSON object follows the readsb trace schema, in which each position report is encoded as the fixed-length array [time\_offset, lat, lon, altitude | “ground”, speed, track, flags, extras]. The absolute timestamp is computed as base\_ts + time\_offset. When the altitude field carries the string “ground”, the record is assigned on\_ground = true and altitude = null. Supplementary per-message fields, including the flight callsign under the “flight” key, are serialized as a JSON string in the extras column.

#### 4.2. Stage 2—Flight Segmentation

The raw Parquet data contains no flight-level entity: messages are partitioned only by callsign, and a single callsign may correspond to multiple distinct flight legs across the observation window. Computing per-flight aggregate features over an unsegmented callsign would conflate the statistical properties of independent departures, producing non-representative context vectors for the anomaly detection models. All downstream feature engineering and anomaly scoring are keyed on flight\_id rather than on callsign. Segment boundaries are detected in adsb-flight-segmentation.ipnyb using a window function partitioned by callsign and ordered by timestamp, producing a binary indicator assigned to 1 under three mutually exclusive conditions, evaluated in priority order:

1. First observation. The record is the first message for the callsign in the dataset.
2. Takeoff transition. The preceding record has on\_ground = true, and the current record has on\_ground = false. This condition is the primary segmentation signal and is immune to receiver-coverage gaps, which produce consecutive records with on\_ground = false on both sides of the gap.
3. Fallback temporal gap. The elapsed time between consecutive messages exceeds 86,400 s (24 h). This condition handles aircraft whose transponders do not broadcast ground state, for which condition (2) would never fire. The threshold of 24 h exceeds the maximum duration of any non-stop commercial service and therefore does not produce false splits for oceanic or polar crossings.

A cumulative sum of the boundary indicator over time within each callsign yields a monotonically increasing segment counter.

#### 4.3. Stage 3—Transform: Feature Engineering

The preprocessing pipeline implemented by adsb-preprocess.ipnyb loads the segmented data and applies a sequence of transformations to produce a standardized 14-dimensional feature vector per record. Domain validity filters then remove records outside physically meaningful bounds: *altitude*  $\in (0, 60,000]$  ft, *speed*  $\in (0, 1,000]$  kn, *heading*  $\in [0^\circ, 360^\circ]$ , *latitude*  $\in [-90^\circ, 90^\circ]$ , *longitude*  $\in [-180^\circ, 180^\circ]$ . Records with null values in any of the six core fields (flight\_id, lat, lon, altitude, speed, heading) are dropped. Three lag features are computed using window functions partitioned by flight\_id and ordered by timestamp. Partitioning by flight\_id prevents lag values from crossing segment boundaries. The first record of each segment with an undefined lag is discarded. altitude\_change and speed\_change are the first differences in their respective fields. heading\_change is the absolute angular difference normalized to the shortest arc in  $[0^\circ, 180^\circ]$  by the transform (2), which correctly handles the circular topology of compass bearings.

$$IF = \begin{cases} 360^\circ - |\Delta h| & \text{if } |\Delta h| > 180^\circ \\ |\Delta h| & \text{if } |\Delta h| \leq 180^\circ \end{cases} \quad (2)$$

Six aggregate statistics are computed per `flight_id` via `groupBy` and joined back to the point-level `DataFrame`: `mean_altitude`, `std_altitude`, `mean_speed`, `std_speed`, `mean_heading_change`, and `std_altitude_change`. Aggregation at the segment level, rather than at the callsign level, ensures that each record's contextual features reflect the statistical properties of the individual flight leg to which it belongs. All 14 features are described in Table 1.

**Table 1.** Feature set used for anomaly detection.

| Feature             | Type         | Description                                                                    |
|---------------------|--------------|--------------------------------------------------------------------------------|
| lat, lon            | Point        | Geodetic coordinates of the position report.                                   |
| altitude            | Point        | Barometric altitude (ft).                                                      |
| speed               | Point        | Ground speed (kn).                                                             |
| heading             | Point        | True track over ground (0–360°).                                               |
| altitude_change     | Lag          | First difference in altitude between consecutive intra-segment messages.       |
| speed_change        | Lag          | First difference in ground speed between consecutive intra-segment messages.   |
| heading_change      | Lag          | Shortest-arc absolute angular difference in heading, normalized to [0°, 180°]. |
| mean_altitude       | Segment agg. | Mean barometric altitude over all airborne points in the segment.              |
| std_altitude        | Segment agg. | Standard deviation of barometric altitude within the segment.                  |
| mean_speed          | Segment agg. | Mean ground speed over the segment.                                            |
| std_speed           | Segment agg. | Standard deviation of ground speed within the segment.                         |
| mean_heading_change | Segment agg. | Mean per-step heading changes magnitude over the segment.                      |
| std_altitude_change | Segment agg. | Standard deviation of per-step altitude change over the segment.               |

The 14-dimensional feature matrix is assembled with `VectorAssembler` and standardized to zero mean and unit variance with `StandardScaler`. Standardization is required because the raw features span different numerical ranges, and both Isolation Forest and GMM are sensitive to feature scale [5–7]. A minimum-segment-length filter discards any segment with fewer than 50 airborne records, as shorter segments do not provide sufficient observations for reliable standard-deviation estimates.

#### 4.3.1. Load Phase—Scaled Features Persistence

The scaled feature `DataFrame` is coalesced and written to `s3a://adsb-data/scaled_features_filtered/`, preserving the following 8 columns: `flight_id`, `flight`, `features` (scaled vector), `lat`, `lon`, `altitude`, `speed`, `heading`, `timestamp`. This materialization decouples the computationally expensive feature engineering from model training. Later stages read from this checkpoint independently and can be executed or re-executed without reprocessing upstream stages.

#### 4.3.2. Stage 5a—Isolation Forest Scoring

At the dataset scale, applying a single trained model to the entire dataset is infeasible due to memory constraints. The pipeline uses Spark's `mapInPandas` API [11] to process the data in distributed batches: the scaled features are repartitioned into approximately 48,209 batches of 100,000 records each, and a separate scikit-learn `IsolationForest` instance is independently trained and scored on each batch within the executor, with no data transferred to the driver during computation. Partitioning is random rather than by `flight_id` so that each batch contains a representative cross-section of flight types, providing a global reference distribution for anomaly boundary estimation. Hyperparameters: `n_estimators` = 100; `contamination` = 0.10; `n_jobs` = −1. A fixed seed was used for reproducibility. These parameters are selected based on fundamental characteristics of the algorithm [5] (p. 6). Specifically, average path lengths converge well before 100 esti-

mators, meaning additional trees would increase the computational overhead without giving proportionate gains in accuracy. The contamination rate of 0.10 was established as a conservative heuristic to isolate the top 10% most extreme structural outliers within the unlabeled ADS-B data. Each record receives a binary label (`is_anomaly`:  $-1$  anomaly,  $+1$  normal) and a continuous `anomaly_score` from `score_samples()`, where lower values denote greater anomalousness.

#### 4.3.3. Stage 5b—Gaussian Mixture Model Scoring

The Gaussian Mixture Model represents the data distribution as a weighted sum of  $K$  multivariate Gaussian components, which are fitted using the Expectation–Maximization algorithm. Anomaly detection is performed by evaluating the log-likelihood of each record under the fitted model: records in low-density regions of the mixture—inconsistent with any fitted component receive low log-probability scores and are classified as anomalous below a percentile threshold. Training the full EM algorithm on 4,820,897,868 records is computationally prohibitive, as each iteration requires a complete dataset pass. The model is instead trained on a randomly sampled 10% subset (482 million rows), which is sufficient for EM convergence to stable parameters. The sample is persisted to disk before training to avoid re-evaluating the sampling lineage on each EM iteration. After training, the model is applied to the full dataset in a single scoring pass using Spark MLlib’s distributed `GaussianMixture.transform()` [12]. Hyperparameters:  $K = 5$  components; maximum 100 EM iterations; convergence tolerance =  $1 \times 10^{-3}$ . These parameters balance performance and computational cost [6].  $K = 5$  captures the main flight phases without overfitting to noise. The iteration limit and tolerance ensure stable convergence on the large training sample, while the 5th percentile threshold sets a conservative boundary for identifying low-density anomalies. The model converged to a final log-likelihood of  $-3,978,576,358.40$ . Fitted component weights are 0.5496, 0.2074, 0.1587, 0.0762, and 0.0081. The anomaly threshold is set at the 5th percentile of the `log_prob_score` distribution ( $-0.2499$ ), labeling the 5% of records with the lowest likelihood as anomalous.

#### 4.3.4. Stage 6—Score Comparison and Ensemble

The Isolation Forest and GMM anomaly scores are on incompatible scales (IF: bounded negative range near  $-0.44$ ; GMM: log-probability down to  $-1.61$ ) and cannot be combined directly. Both distributions are independently normalized to z-scores as  $z = (score - \mu) / \sigma$ , where  $\mu$  and  $\sigma$  are computed from a single SQL scan over each deduplicated dataset. After normalization, the IF distribution has  $\mu = -0.0004$ ,  $\sigma = 1.0003$ , and the GMM distribution has  $\mu = -0.3052$ ,  $\sigma = 1.9404$ ; in both cases, a more negative z-score indicates greater anomalousness. Prior to joining, each scored dataset is deduplicated on (`flight_id`, `timestamp`), removing 374,686 duplicate records from the IF output and 2459 from the GMM output. The two datasets are joined on (`flight_id`, `timestamp`) by an inner join, yielding 482,480,432 aligned records used for cross-model comparison. An important distinction in the comparison is that the two models operate at different granularities. Isolation Forest scores each record independently: every point receives its own anomaly score based on its position in the 14-dimensional feature space. GMM, by contrast, evaluates the log-probability of the flight-level feature vector; when the scored output is joined back to the point-level data, every record within the same flight segment receives the same GMM score. Consequently, a flight that GMM considers anomalous assigns a uniformly low GMM z-score to all of its constituent points, whereas Isolation Forest may flag only specific records within that flight, such as those corresponding to unusual climb rates, abrupt heading changes, or transition phases. This granularity mismatch explains much of the observed score disagreement: within a GMM-anomalous flight, structurally routine cruise points

will show large positive  $\Delta z$  (IF normal, GMM anomalous), while operationally unusual manoeuvre points may show close agreement. Conversely, a GMM-normal flight may contain isolated IF-anomalous points that are locally atypical but consistent with the flight's overall probability density. The ensemble score is the arithmetic mean of the two z-scores. Averaging two weakly correlated signals (Pearson  $r = 0.2271$ ) partially reduces variance: the ensemble distribution ( $\sigma \approx 1.188$ ) is narrower than the GMM distribution ( $\sigma \approx 1.94$ ), though not narrower than the IF distribution ( $\sigma \approx 1.0$ ). The combined dataset, including both individual z-scores and the ensemble score, is written to `s3a://adsb-data/anomalies/combined_anomaly_scores/`.

#### 4.4. Experimental Evaluation

##### 4.4.1. Dataset

The dataset comprises 4,820,897,868 ADS-B records produced by the pipeline after flight segmentation and feature engineering. The input to both anomaly detection models is the filtered, scaled\_features\_filtered Parquet dataset, which retains only airborne records with segments of at least 100 points. The raw corpus before preprocessing contained 1,192,865,697 records spanning 67 daily partitions (1 January 2026 through 8 March 2026); the post-segmentation scaled dataset is larger because each flight leg produces its own feature rows with independently computed segment-level aggregates. Table 2 summarizes the descriptive statistics of the five core flight parameters measured across the filtered dataset.

**Table 2.** Descriptive statistics of core ADS-B features after preprocessing.

| Parameter     | Mean   | Std Dev | Min     | Max    | Count         |
|---------------|--------|---------|---------|--------|---------------|
| Latitude (°)  | 32.82  | 20.03   | −89.95  | 89.99  | 4,830,293,182 |
| Longitude (°) | −38.11 | 81.04   | −180.00 | 180.00 | 4,830,293,182 |
| Altitude (ft) | 14,448 | 13,930  | 0       | 60,000 | 4,830,293,182 |
| Speed (kn)    | 264.97 | 149.18  | 0       | 1000   | 4,830,293,182 |
| Heading (°)   | 183.86 | 103.06  | 0       | 359.9  | 4,830,293,182 |

##### 4.4.2. Experimental Setup

All experiments were conducted on a single-host Apache Spark 3.5.0 standalone cluster with 32 CPU cores. Table 3 summarizes the runtime configuration per pipeline stage. Adaptive Query Execution (AQE) and Kryo serialization are enabled throughout. The preprocessing stage allocates all 32 cores to a single executor; the Isolation Forest and GMM stages each use a single 32-core executor with 72 GB of executor memory to accommodate the in-memory batch model fitting and the full-dataset scoring pass, respectively.

**Table 3.** Spark runtime configuration for each pipeline stage.

| Parameter              | Preprocessing | Isolation Forest | GMM     |
|------------------------|---------------|------------------|---------|
| Spark version          | 3.5.0         | 3.5.0            | 3.5.0   |
| Driver memory          | 16 GB         | 12 GB            | 12 GB   |
| Executor memory        | 48 GB         | 60 GB            | 60 GB   |
| Executor cores         | 32 (all)      | 10               | 10      |
| Max cores              | 64            | 30               | 30      |
| SQL shuffle partitions | 64            | 60               | 60      |
| Adaptive query exec.   | Enabled       | Enabled          | Enabled |
| Serializer             | Kryo          | Kryo             | Kryo    |
| Total CPU cores        | 32            | 32               | 32      |
| Wall-clock time        | 3.9 h         | 1.7 h            | 4.8 h   |

#### 4.4.3. Comparative Results

Table 4 presents the quantitative comparison of the two models on the full 4,830,293,182-record dataset. Raw scores are on incompatible scales (IF: bounded range  $[-0.775, -0.319]$ ; GMM: log-probability range  $[-1.609, 0.000]$ ) and are normalized to z-scores prior to ensemble computation, as described in Section 4.5.

**Table 4.** Performance comparison of Isolation Forest and Gaussian Mixture Model.

| Metric                      | Isolation Forest   | GMM               | Notes                     |
|-----------------------------|--------------------|-------------------|---------------------------|
| Records processed           | 4,820,897,868      | 4,820,897,868     | Same input                |
| Anomalies detected          | 482,105,746        | 240,091,382       | Different thresholds      |
| Anomaly rate                | 10.00%             | 4.98%             | contamination vs. 5th pct |
| Raw score mean              | −0.5318            | −0.0348           | Incompatible scales       |
| Raw score std dev           | 0.0344             | 0.1239            | GMM wider spread          |
| Raw score range             | $[-0.768, -0.386]$ | $[-1.609, 0.000]$ | -                         |
| Training time (wall clock)  | ~82 min            | ~259 min          | GMM 3 times slower        |
| Inference time (wall clock) | ~21 min            | ~28 min           | Similar at scale          |
| Total pipeline time         | ~1.7 h             | ~4.8 h            | -                         |

Z-score distribution statistics computed on the 482,480,432-record aligned dataset (the subset of records flagged as anomalous by IF) are presented in Table 5. The IF z-score distribution on the anomalous subset has mean =  $-0.0004$ , std = 1.0003; the GMM z-score distribution has a heavily right-skewed shape (mean =  $-0.3052$ , std = 1.9404) with a compressed upper tail (P25 through P75 spanning only 0.077 units), reflecting that the majority of records cluster tightly near the maximum log-probability while a sparse left tail extends to  $-12.708$ . The ensemble z-score distribution (std = 1.1882) exhibits reduced variance relative to either individual model [13], consistent with the variance-reduction effect of averaging two weakly correlated signals (Pearson  $r = 0.227$ , classified as weak positive correlation).

**Table 5.** Score agreement categories on the records where IF and GMM scores are aligned.

| Score            | Mean    | Std    | Min     | P25    | Median | P75   | Max   |
|------------------|---------|--------|---------|--------|--------|-------|-------|
| IF z-score       | 0.0004  | 1.0003 | −6.877  | −0.499 | 0.270  | 0.753 | 4.254 |
| GMM z-score      | −0.3052 | 1.9404 | −12.708 | 0.204  | 0.279  | 0.281 | 0.281 |
| Ensemble z-score | −0.1528 | 1.1882 | −9.792  | −0.315 | 0.151  | 0.477 | 2.250 |

The Pearson correlation between IF z-scores and GMM z-scores is  $r = 0.231$  (weak positive). The top-100 anomaly lists of the two models share 0% overlap, confirming that the methods identify structurally distinct populations of anomalous records. Isolation Forest detects records with anomalously short isolation path lengths—structural outliers in the 14-dimensional feature space. GMM detects records whose feature combinations fall in low-density regions of the fitted mixture, capturing contextual anomalies that may not be structural outliers but are inconsistent with any of the five fitted normal-behavior components. The label-agreement breakdown across the 4,830,293,182 aligned records is presented in Table 6.

Table 6 shows five  $\Delta z$ -agreement categories. The first three rows (Close 38.19%, Moderate 31.01% and Strong Disagreement 30.80%) partition the full 482M aligned records by z-score difference magnitude. The last two rows (Disagree on High 10.08%, Both High Anomaly 1.32%) are a supplementary breakdown by the  $z < -2$  extreme threshold—note these overlap with rows 1–3 rather than being mutually exclusive with them, so the five rows intentionally sum to more than 100%.

**Table 6.** Score agreement categories on the records where IF and GMM scores are aligned.

| Agreement Category                            | Record Count | Percentage |
|-----------------------------------------------|--------------|------------|
| Close Agreement ( $ \Delta z  < 0.5$ )        | 184,259,013  | 38.19%     |
| Moderate Agreement ( $ \Delta z  < 1.0$ )     | 149,638,736  | 31.01%     |
| Strong Disagreement ( $ \Delta z  \geq 1.0$ ) | 148,582,683  | 30.80%     |
| Disagree on High Anomaly                      | 48,614,476   | 10.08%     |
| Both High Anomaly (both $z < -2$ )            | 6,377,482    | 1.32%      |

#### 4.4.4. Top Detected Anomalies

The most anomalous flight segments, ranked by minimum ensemble z-score, are dominated by records with missing or malformed callsign fields: blank identifiers, the single-character callsign “N”, and the all-zeros string “00000000”. These records systematically exhibit atypical feature combinations, and in particular, degenerate segment-level aggregate statistics arising from the absence of a consistent flight identity, and are correctly identified as the highest-confidence anomalies by both models, with IF z-scores reaching  $-6.877$  and GMM z-scores reaching  $-12.708$ , yielding ensemble z-scores as low as  $-9.792$ . Beyond data-quality anomalies, the two models surface different categories of operationally meaningful anomalies. Isolation Forest flags records with extreme structural feature values: transponder data corruptions (callsign “@@@@@@”), non-standard VFR callsigns, and airline flights with trajectory characteristics inconsistent with the population distribution at the time of scoring. GMM identifies records whose feature combinations fall outside the five fitted Gaussian components: predominantly general aviation aircraft (characterized by low altitude, low speed, and irregular heading changes) whose flight profiles occupy low-density regions of the mixture that are not represented by any of the dominant commercial-cruise components. The flight segment identified by both models as anomalous (“MUSL”) represents the highest-confidence operationally genuine anomaly in the dataset, simultaneously exhibiting a short isolation path length and a low mixture log-probability. The ensemble score provides more conservative coverage than either individual model: the ensemble  $|z| > 3$  threshold yields fewer flagged records than either IF or GMM alone (18,069,894 vs. 24,002,870 and 37,366,570, respectively), reflecting that the ensemble selects records that are simultaneously detected as anomalous from both structural and probabilistic perspectives.

#### 4.5. Discussion

The observed low correlation ( $r = 0.227$ ) between the anomaly scores of the Isolation Forest (IF) and the Gaussian Mixture Model (GMM) suggests that the two algorithms are capturing fundamentally different types of data irregularities. While IF effectively isolates point anomalies via recursive partitioning of the feature space, GMM identifies deviations based on the underlying probability density. This divergence justifies an ensemble approach. By combining both scores, the system can achieve a more holistic detection profile that captures structural outliers and statistical deviations that a single model might miss. From a computational perspective, the Isolation Forest demonstrated excellent scalability, processing the dataset significantly faster than the GMM ( $\sim 1.7$  h vs.  $\sim 4.8$  h). This is attributed to the linear time complexity of IF and its lack of iterative convergence requirements, unlike Expectation–Maximization used in GMM. For real-time aviation monitoring systems where low latency is critical, IF provides a more viable baseline, whereas GMM might be reserved for offline, high-precision forensic analysis of flight data. The detection of anomalies in features such as vertical rate and ground speed highlights the potential for this pipeline to identify unsafe flight configurations or sensor malfunctions. However, a primary challenge remains the distinction between ‘benign’ anomalies (e.g., extreme

weather maneuvers or rare but safe flight paths) and ‘malignant’ anomalies (e.g., ADS-B spoofing or mechanical failure). Future iterations of this work should involve domain experts to label a subset of the detected anomalies, allowing for a transition from purely unsupervised detection to a semi-supervised validation framework. The scalability of the Spark-based ETL pipeline proved essential given the volume of nearly 5 billion records. However, the models’ sensitivity to Z-score normalization, particularly GMM, suggests that data preprocessing remains the most critical stage. The presence of ‘ground’ flags and null values in raw ADS-B telemetry requires a robust imputation strategy to prevent models from flagging missing data as false positives.

#### 4.6. Conclusions

This study demonstrates the feasibility and effectiveness of using distributed machine learning for large-scale anomaly detection in ADS-B air traffic data. By implementing an ETL-based pipeline on an Apache Spark cluster, a massive dataset of approximately 4.83 billion records was successfully processed. The comparative analysis of two unsupervised methods shows that:

- **Complementary Detection Profiles:** The low Pearson correlation ( $r \cong 0.227$ ) between the two models suggests that they capture distinct types of anomalies. While IF excels at isolating sparse, structurally atypical observations via recursive partitioning, GMM provides a probabilistic framework for modeling multimodal “normal” behaviors.
- **Scalability:** The pipeline’s architecture, using Spark’s mapInPandas for IF and a 10% sampling strategy for GMM training, enabled analysis of billion-scale datasets within reasonable wall-clock times ( $\sim 1.7$  h for IF and  $\sim 4.8$  h for GMM).
- **Operational Utility:** Using an ensemble score reduces variance and provides a more robust indicator of abnormality, which is critical for reducing false alarms in high-stakes aviation monitoring environments.

Despite these results, several challenges remain. The current approach is purely unsupervised, meaning that while statistical anomalies are identified, their operational significance (e.g., distinguishing between a sensor error and a genuine safety risk) requires expert domain validation. Future work should focus on incorporating a labeled subset of historical incidents to transition toward semi-supervised learning models; extending the current batch-processing pipeline to a streaming architecture using Spark Structured Streaming [14] for real-time surveillance; and including meteorological data and airspace constraints to provide better context for the detected anomalies.

In summary, lightweight unsupervised methods such as IF and GMM, when deployed within a distributed computing framework, provide a scalable and effective baseline for enhancing the safety and integrity of global air traffic monitoring systems.

**Author Contributions:** Conceptualization, R.F.; methodology, M.B. and R.F.; software, M.B.; validation, M.B.; formal analysis, M.B. and R.F.; investigation, M.B.; resources, R.F. and M.S.; data curation, M.B. and R.F.; writing—original draft preparation, M.B. and R.F.; writing—review and editing, M.S. and R.F.; visualization, M.B.; supervision, R.F.; project administration, M.S.; funding acquisition, M.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the European Regional Development Fund within the OP “Research, Innovation and Digitalization Programme for Intelligent Transformation 2021–2027”, Project No. BG16RFPR002-1.014-0005 Center of Competence “Smart Mechatronics, Eco- and Energy Saving Systems and Technologies”.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The ADS-B data used in this study are publicly available through the ADSB.lol open data project at <https://github.com/adsblol> (accessed on 1 March 2026). The code developed for anomaly detection, including preprocessing scripts and model implementations, is openly accessible at <https://github.com/dinozavyr/ADS-B-anomaly-detection> (accessed on 1 March 2026).

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

## Abbreviations

The following abbreviations are used in this manuscript:

|       |                                            |
|-------|--------------------------------------------|
| ADS-B | Automatic Dependent Surveillance-Broadcast |
| API   | Application Programming Interface          |
| AQE   | Adaptive Query Execution                   |
| GMM   | Gaussian Mixture Model                     |
| IF    | Isolation Forest                           |
| EM    | Expectation–Maximization                   |
| ETL   | Extract Transform Load                     |
| SQL   | Structured Query Language                  |
| JSON  | JavaScript Object Notation                 |
| VFR   | Visual Flight Rules                        |

## References

1. Zhang, J.; Liu, W.; Zhu, Y. Study of ADS-B Data Evaluation. *Chin. J. Aeronaut.* **2011**, *24*, 461–466. [CrossRef]
2. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A Survey. *ACM Comput. Surv.* **2009**, *41*, 58. [CrossRef]
3. Jasra, S.; Gauci, J.; Muscat, A.; Valentino, G.; Zammit-Mangion, D.; Camilleri, R. Literature review of machine learning techniques to analyse flight data. In *Advanced Aircraft Efficiency in a Global Air Transport System*; Association Aéronautique et Astronautique de France: Toulouse, France, 2018. Available online: <https://www.um.edu.mt/library/oar/handle/123456789/58839> (accessed on 1 March 2026).
4. Coelho e Silva, L.; Murça, M.C.R. Machine learning Models for Online Anomaly Detection in Flight Operations. In Proceedings of the AIAA AVIATION Forum 2023, San Diego, CA, USA, 12–16 June 2023. [CrossRef]
5. Liu, F.T.; Ting, K.M.; Zhou, Z.-H. Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM), Pisa, Italy, 15–19 December 2008; pp. 413–422. [CrossRef]
6. Li, L.; Hansman, R.J.; Palacios, R.; Welsch, R. Anomaly detection via a Gaussian Mixture Model for flight operation and safety monitoring. *Transp. Res. Part C Emerg. Technol.* **2016**, *64*, 45–57. [CrossRef]
7. Liu, F.T.; Ting, K.M.; Zhou, Z.-H. Isolation-Based Anomaly Detection. *ACM Trans. Knowl. Discov. Data* **2012**, *6*, 3. [CrossRef]
8. Munir, M.; Chattha, M.A.; Dengel, A.; Ahmed, S. A Comparative Analysis of Traditional and Deep Learning-Based Anomaly Detection Methods for Streaming Data. In Proceedings of the 2019 18th IEEE International Conference On Machine Learning and Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019; pp. 561–566. [CrossRef]
9. Azz, M.E.A.; Aljasmi, A.; Fallah Seghrouchni, A.E.; Benzarti, W.; Chopin, P.; Barbaresco, F.; Zitar, R.A. ADS-B Data Anomaly Detection with Machine Learning Methods. In Proceedings of the 2024 11th International Workshop on Metrology for AeroSpace (MetroAeroSpace), Lublin, Poland, 3–5 June 2024; pp. 94–99. [CrossRef]
10. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830. Available online: <https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf> (accessed on 1 March 2026).
11. Zaharia, M.; Xin, R.S.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Meng, X.; Rosen, J.; Venkataraman, S.; Franklin, M.J.; et al. Apache Spark: A Unified Engine for Big Data Processing. *Commun. ACM* **2016**, *59*, 56–65. [CrossRef]
12. Meng, X.; Bradley, J.; Yavuz, B.; Sparks, E.; Venkataraman, S.; Liu, D.; Freeman, J.; Tsai, D.B.; Amde, M.; Owen, S.; et al. Mllib: Machine learning in Apache Spark. *J. Mach. Learn. Res.* **2016**, *17*, 1–7. Available online: <https://www.jmlr.org/papers/volume17/15-237/15-237.pdf> (accessed on 1 March 2026). [CrossRef]

13. Dietterich, T.G. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1857. [[CrossRef](#)] [[PubMed](#)]
14. Armbrust, M.; Das, T.; Torres, J.; Yavuz, B.; Zhu, S.; Xin, R.; Ghodsi, A.; Stoica, I.; Zaharia, M. Structured Streaming: A Declarative API for Real-Time Applications in Apache Spark. In Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18), Houston, TX, USA, 10–15 June 2018; pp. 601–613. [[CrossRef](#)]

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