Satellite Pattern-of-Life Identification Challenge: Competition Design and Results

Peng Mun Siew[*], **Haley E. Solera**[†], **Giovanni Lavezzi**[‡], **Thomas G. Roberts**[§], **Daniel Jang**[¶]
Massachusetts Institute of Technology

David Baldsiefen[‖]
Julius-Maximilians-University of Würzburg

Binh Tran, **Christopher Yeung**, **Kurtis Johnson**, **Nathan Metzger** [**]
Millennial Software Solutions Inc. & Indiana University of Pennsylvania

Francois Porcher, **Isaac Haik** ,[††]
University of California, Berkeley & BNP Paribas

Victor Rodriguez-Fernandez,[‡‡]
Universidad Politécnica de Madrid

Jeffrey Price,[§§]
United States Air Force

Jonathan P. How,[¶¶] **Richard Linares**[***]
Massachusetts Institute of Technology

ABSTRACT

Artificial intelligence (AI) techniques have emerged as powerful tools for solving complex problems across various domains. Despite the availability of extensive historical data on Earth-orbiting objects, the adoption of AI into the space domain awareness (SDA) community remains limited. To address this gap, the Massachusetts Institute of Technology (MIT) launched the 2024 MIT ARCLab Prize for AI Innovation in Space, aimed at leveraging AI to characterize satellite pattern-of-life (PoL) in Geostationary Earth Orbit (GEO). The challenge involves analyzing multivariate time-series data, identifying behavioral patterns, and detecting anomalies to promote a safer and sustainable space environment. The challenge dataset comprises 2402 satellite trajectories collected over six months with a two-hour temporal resolution. The data include simulated trajectories, Vector Covariance Message (VCM) data, and two-line elements (TLE), all generated using high-fidelity satellite propagators. This dataset features diverse operational behaviors and propulsion systems, providing a robust foundation for AI analysis. Participants submitted their AI models on EvalAI platform, evaluated on precision, recall, and technical write-up. The competition attracted over 100 teams globally, comprising students, machine learning experts, aerospace engineers, and industry professionals, resulting in more than 350 submissions. This paper discusses the results of the successful AI SDA challenge, where Team Hawaii2024 emerged as the winner, Team Millennial-IUP as the runner-up, and Team QR_Is as the second runner-up, with the top F_2 scores reaching 0.952 on the partial test set. The results highlight the AI's ability to automate data analysis and uncover patterns in satellite behavior. The results provide valuable insights into state-of-the-art AI techniques for behavioral pattern recognition in SDA.

[*]Corresponding author. Research Assistant, Department of Aeronautics and Astronautics, *siewpm@mit.edu*
[†]Ph.D. Student, Department of Aeronautics and Astronautics, *hsolera@mit.edu*
[‡]Postdoctoral Associate, Department of Aeronautics and Astronautics, *glavezzi@mit.edu*
[§]Ph.D. Student, Department of Aeronautics and Astronautics, *thomasgr@mit.edu*
[¶]Technical Staff, MIT Lincoln Laboratory, *djang@mit.edu*
[‖]Team Hawaii2024, *david.baldsiefen@outlook.de*
[**]Team Millennial-IUP, *btran@millennial.software*, *cyeung@millennial.software*, *kjohnson@millennial.software*, *nmetzger@millennial.software*
[††]Team QR_Is: *francois_porcher@berkeley.edu*, *isaac.haik@gmail.com*
[‡‡]Associate Professor, Department of Computer Systems Engineering, *victor.rfernandez@upm.es*
[§§]Director of Technology Management, *pricej@mit.edu*
[¶¶]Professor, Department of Aeronautics and Astronautics, *jhow@mit.edu*
[***]Associate Professor, Department of Aeronautics and Astronautics, *linaresr@mit.edu*

1. INTRODUCTION

In recent years, the application of artificial intelligence (AI) techniques to solve complex problems has received significant attention within the scientific community. However, there remains a notable lack of adoption of these techniques within the space domain awareness (SDA) community. Despite the wealth of historical data available on Earth-orbiting objects, including orbital state and light curve measurements, the disconnect between the AI and SDA research communities has hindered interdisciplinary progress. To bridge this gap, the Massachusetts Institute of Technology (MIT) has launched the 2024 MIT ARCLab Prize for AI innovation in Space focusing on leveraging AI algorithms to characterize satellite pattern-of-life (PoL) within the Geostationary Earth Orbit (GEO) [1]. GEO satellites are crucial for various applications, including communication, weather monitoring, guidance and navigation, and Earth observation. Throughout their operational lifetime, these satellites exhibit a range of behavioral modes, such as station-keeping, longitudinal shifts, and end-of-life behaviors. Satellite PoLs, which describe sequences of both natural and non-natural behaviors modes, are essential for comprehending these on-orbit activities.

This unique challenge combines complex multivariate time-series data analysis, behavioral pattern recognition, and anomaly detection to enhance space safety and sustainability. One of the primary challenges is the variability in GEO satellite behaviors, which can be attributed to factors such as the lack of standard operational guidelines, differing mission objectives, and diverse propulsion systems. This variability complicates the manual identification and classification of satellite behaviors, underscoring the need for advanced analytical tools.

AI algorithms are adept at recognizing and learning from patterns, making them particularly well-suited for processing SDA data. The increasing number of space objects and the vast volume of data generated daily make manual analysis increasingly challenging. AI algorithms can alleviate this burden by automating repetitive and mundane tasks, freeing up human resources for more critical decision-making. Additionally, advanced AI algorithms can predict future behaviors and assist in proactive decision-making. AI algorithms can also help uncover complex patterns and non-intuitive relationships that are obscured within the noisy time-series data.

The challenge dataset includes 2402 satellite trajectories over six months, with a two-hour temporal resolution. These trajectories include 2000 simulated data points, 106 from Vector Covariance Message (VCM) data, and the remaining 296 generated from two-line elements (TLE) data. The synthetic data was generated using an in-house satellite simulation tool developed in collaboration with the MIT Lincoln Laboratory. This tool utilizes a high-fidelity satellite propagator to simulate the trajectories of satellites under various operating objectives and propulsion capabilities. Each simulated satellite trajectory features satellites with unique physical parameters, initial orbital elements, and propulsion systems. The dataset is divided into training, public test, and private test sets. A private test set is used to prevent the participants from over-fitting to the public test set. Furthermore, trajectories from different periods are included in the test data to evaluate the generalization capability of the algorithms. Accompanying the challenge dataset is a development toolkit that includes basic utility functions for data parsing, manipulation, and visualization, as well as baseline heuristic and machine learning solutions* [2].

The challenge was hosted on the EvalAI platform†, where participants submitted trained models and algorithms to be evaluated on a hidden test dataset [3]. Submissions were assessed based on precision and recall scores using the F_2 metric, and the quality of their technical write-up, with an 80:20 weighting. Technical write-ups were evaluated by a panel of experts on clarity, novelty, technical depth, reproducibility, and insights. This comprehensive evaluation ensured that both the algorithm's performance and the value of technical insights are recognized and rewarded in the competition.

The AI SDA challenge was successful in that it attracted over 100 teams from diverse backgrounds and regions worldwide. These teams have collectively submitted more than 350 entries on our evaluation platform, bringing a diverse range of models, including convolution neural networks, long short-term memory networks, random forests, transformers, gradient boosting algorithms, recurrent neural networks, heuristic models, and various hybrid approaches. An encouraging outcome of the challenge was the high level of performance achieved by many teams, with the majority of submissions surpassing the baseline solution provided. The top teams—Team Hawaii2024 as the winner, Team Millennial-IUP as the first runner-up, and Team QR_Is as the second runner-up—achieved F_2 scores greater than 0.9 on the partial test set and over 0.8 on the more challenging full test set.

*https://splid-devkit.readthedocs.io/en/latest/README.html
†https://eval.ai/web/challenges/challenge-page/2164/overview

Analysis of trends and patterns among the submissions provides valuable insights into the current state-of-the-art in AI techniques for behavioral pattern recognition in SDA. The methodology developed by the top three teams are presented in detail. Additionally, this paper discusses the implications of these findings for future research and development efforts in the field, emphasizing the importance of continued innovation and collaboration between the AI and SDA communities.

2. THE CHALLENGE PROBLEM

The challenge addresses the critical need for more efficient and accurate tracking and orbit prediction capabilities for active satellites within the increasingly congested near-Earth space environment. As space activities continue to proliferate, the demand for advanced technologies to effectively monitor and manage satellite behavior has become paramount. This challenge aims to provide a solution by developing cutting-edge AI algorithms for automated change detection and behavior classification using astrometric time-series data and satellite PoLs.

Throughout their operational lifetimes, Geostationary Earth Orbit (GEO) satellites exhibit a variety of behaviors and behavior patterns, including station-keeping, longitudinal shift maneuvers, end-of-life behaviors, and natural drift. Satellite PoLs describe these on-orbit behaviors as sequences of both natural and non-natural behavioral modes.

Participants in this challenge were tasked with creating models that can accurately label and timestamp the behavioral modes of GEO satellites over a six-month period. The performance of these AI algorithms will be evaluated against an internal test dataset. The ultimate objective is to advance the state-of-the-art in satellite PoL characterization, enhance time-series analysis capabilities, and inspire innovative solutions with broad applicability across multiple domains.

2.1 Dataset

Participants were provided with a public dataset containing 1900 satellite state histories and their associated PoL labels. Competitors were also permitted to augment these state histories as a means of supplementing and expanding their training inputs. A private test set consisting of 502 state histories and their truth labels was used to evaluate each team's top-performing algorithm. Both the public and the private datasets included simulated and historical satellite PoLs and were labelled according to stringent conventions to ensure consistency across multi-source data.

2.1.1 PoL Labels

Each state history in the challenge dataset consists of the osculating orbital elements, geodetic positions, and the Cartesian position and velocity in the J2000 inertial reference frame over six months at a two-hour time step. Their corresponding labels reflect two specific features of interest: periods of station-keeping and the type of propulsion used to perform nominal station-keeping maneuvers. To that end, there are four behavioral mode classes represented in the challenge dataset—not station-keeping (NK), station-keeping with electric propulsion (EK), station-keeping with chemical propulsion (CK), and station-keeping with hybrid propulsion (HK). Every time step within a satellite's PoL is associated with one of these behavioral modes.

When a satellite transitions from one behavioral mode to another, the first time step within the new behavioral mode is designated as a PoL node and can be labeled according to the type of transition for clarity or to facilitate subsequent labeling tasks. In this case, PoL nodes are identified by one of three labels—initiate drift (ID), adjust drift (AD), or initiate station-keeping (IK)—as a means of delineating the change detection aspect of the challenge problem from that of behavior classification. In effect, competing algorithms must detect and identify any PoL nodes present within a satellite's state history and assign a class label to each demarcated behavioral mode.

Additionally, these tasks are to be completed for both the longitudinal (east-west) and transverse (north-south) directions in parallel since many station-keeping characteristics and tell-tale operational shifts are directionally isolated or distinguished. PoL labels are therefore tabulated in columns, starting with the time index of each node, followed by the direction (EW or NS), the node label, and finally the type of behavioral mode proceeding from the specified time index.

2.1.2 Public Dataset

The public challenge dataset was comprised primarily of synthetic state histories that were generated using a simulation tool developed at the MIT Lincoln Laboratory. All simulated trajectories were propagated using a high-fidelity special perturbations model and sampled at two-hour time steps. Each simulation featured a unique combination of parameters and control logic designed to mimic a variety of historical GEO satellite behaviors. The three propulsion types were distinguished by impulsive and non-impulsive maneuver profiles and through parameters such as maneuver frequency and consistency which were chosen to reflect a range of thruster configurations common to each category.

Truth labels were generated autonomously and reviewed manually in batches by a subject-matter expert. The placement of each PoL node was determined by the time step of the effective shift in control logic instead of any corresponding maneuver. In particular, this allowed for multiple simulated ID node triggers, including both intentional longitudinal shift maneuvers and unintentional loss of station. Every category of PoL node and behavioral mode was represented within the simulated data, as were diverse combinations of east-west and north-south behaviors. In total, there were 1870 synthetic state histories in the publicly-released dataset.

The remaining 30 objects in the training dataset originated from historical VCMs supplied by the United States Joint Space Operations Center (JSpOC) and were manually labelled by a subject-matter expert. Since VCM states are updated less frequently than the simulated state histories, the VCM-derived trajectories are interpolated to the standard two-hour time step. Overall, the public dataset included 1900 distinct PoLs representing a total of 5,355 nodes and 8,957 behavioral modes. Table 1 contains the distribution of these features amongst the label categories.

Table 1: Distribution of PoL labels within the private and public datasets.

	Mode Type	**Private Qty**	**Public Qty**	**Node Label**	**Private Qty**	**Public Qty**
SIM	NK	374	5255	ID	116	1812
	CK	124	1558	IK	165	2154
	HK	44	582	AD	100	1374
	EK	99	1487			
VCM	NK	43	23	ID	6	3
	CK	65	27	IK	12	7
	HK	38	7	AD	13	5
	EK	35	18			
TLE	NK	144	0	ID	36	0
	CK	185	0	IK	36	0
	HK	118	0	AD	32	0
	EK	247	0			
Total	NK	561	5278	ID	158	1815
	CK	374	1585	IK	213	2161
	HK	200	589	AD	145	1379
	EK	381	1505			

2.1.3 Private Evaluation Dataset

The evaluation dataset contains 502 state histories, of which 130 are simulated, 76 are derived from VCMs, and 296 are derived from TLEs. All VCM and TLE-derived ephemerides are interpolated to a two-hour time step using a special perturbation propagator configured in Poliastro [4]. This method utilizes Cowell's formulation and accounts for the J2 perturbation as well as the third body effects of the Sun and Moon. All historical PoLs were labelled by a subject-matter expert using the same conventions applied to the public dataset.

In **Phase I**, the competing algorithms were evaluated on 32 VCM-derived and 130 synthetic state histories from the private dataset to generate a real-time public leaderboard on EvalAI. The remaining 340 state histories were not used to avoid participants overfitting to the full evaluation dataset. At the end of **Phase I**, all submissions were evaluated

on the full 502 PoLs in the private dataset, which included 516 PoL nodes and 1,516 behavioral modes, and resulted in the scores displayed in Table 3. Table 1 details the distribution of PoL features within the private evaluation dataset.

2.2 Challenge Phases and Evaluation Metrics

The challenge is divided into two distinct phases—**Phase I** and **Phase II**. During **Phase I**, participants are ranked solely based on the performance of their submitted model. The evaluation emphasizes the model's ability to accurately detect and classify satellite behavioral mode changes in comparison to the ground truth labels. The primary evaluation metric is the F_2 score, which places greater weight on recall than precision. Precision represents the proportion of correctly detected nodes among all detected nodes, whereas recall indicates the percentage of actual true nodes that were correctly identified. The formulas for precision, recall, and the F_2 score are defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \quad F_2 = \frac{5 \cdot \text{Precision} \cdot \text{Recall}}{4 \cdot \text{Precision} + \text{Recall}} \tag{1}$$

where the terms true positive (TP), false positive (FP), and false negative (FN) are defined in table 2.

Table 2: Description of Evaluation Metrics.

Acronym	Description
TP	The participant node falls within the time tolerance interval of the ground truth node, and the labels match.
FP	The participant node falls within the time tolerance interval of a ground truth node, but the labels do not match. **or** There are no ground truth nodes within the tolerance interval around the participant node.
FN	Missed ground truth node; there is no participant node within the tolerance interval of a ground truth node.

In the event of a tie in the F_2 scores, the final ranking will factor in the timing accuracy of correct assignments, where the final metric will be adjusted based on the temporal proximity between the correctly predicted and the actual node, with more accurate timing yielding a higher score.

The top 10 participants from the **Phase I** full test set leaderboard are invited to participate in **Phase II**. During this phase, participants will submit a technical report detailing their AI algorithms. The evaluation in **Phase II** is based on a composite score (CS) that incorporates both the algorithm's performance and the quality of the technical report. The composite score is calculated as follows:

$$\text{CS} = 0.8 \cdot F_{2,\text{norm}} + 0.2 \cdot Q \tag{2}$$

where $F_{2,\text{norm}}$ represents the F_2 metric normalized to a maximum value of 1, with 1 corresponding to the highest F_2 score achieved on the full test set leaderboard. The quality score Q assesses the innovation and overall quality of the technical report, including clarity, novelty, technical depth, reproducibility, and insights. The technical reports are independently evaluated by a panel of expert judges on a 0-5 scale for each criterion, with higher scores indicating better quality. These scores are then summed, averaged across the judges, and divided by 25 (the maximum possible score) to produce Q, resulting in a normalized score between 0 and 1. This evaluation process ensures that both the precision of the algorithms and the quality of the technical documentation are recognized and rewarded.

3. CHALLENGE RESULTS

The 2024 AI Challenge attracted substantial interest, with 126 teams registering on the challenge website. Of these, 35 teams actively participated, collectively making over 367 successful submissions on the EvalAI platform. The challenge drew a diverse group of participants from various countries and professional backgrounds, underscoring a global interest in AI and its application. The majority of participants were from the USA and India, followed by Australia, New Zealand, Portugal, Germany, Hong Kong SAR, China and, several other nations.

Participants are predominantly from academia or industry, with 52.5% of the participants were affiliated with academic institutions. This significant academic presence suggests a strong interest in research and educational aspects of AI. Meanwhile, 37.3% of participants were from the industry sector, highlighting the practical and commercial applications of AI that attract professional interest. Furthermore, there were representatives from government, retired professionals, non-profit organizations, and independent practitioners, adding to the diversity of perspectives and expertise. A significant portion of participants were first time participants, with 76.3% of the participants indicated no prior AI challenge experience.

The competition elicited a diverse range of model submissions, including convolutional neural networks (CNNs), long short-term memory networks (LSTMs), random forests, transformers, gradient boosting algorithms like XGBoost and LightGBM, recurrent neural networks (RNNs), heuristic models, and various hybrid approaches. Notably, over two-thirds of the participating teams opted to utilize separate models for the east-west and north-south PoL features. Many teams incorporated additional input features derived from statistical measures such as the standard deviation, mean, maximum, and minimum values of various signals within a fixed time window. The majority of machine learning approaches leveraged lagged features, with a common lag range of 4-6 steps. No single model or approach dominated the leaderboard, though CNNs, RNNs, and random forest methods were frequently represented among the top-performing solutions.

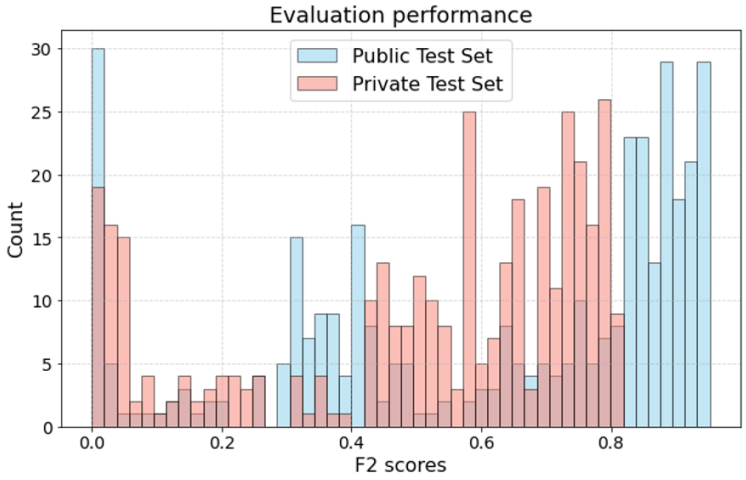


Fig. 1: Algorithm performance displayed as F2 score for the partial (blue) and full (orange) test sets.

As shown in Figure 1, the performance of the finalist teams varied across the partial and full evaluation datasets. In the partial test set, finalists were able to detect between 58% and 87% of nodes and classify between 31% to 69% of station-keeping behavioral modes correctly. However, when evaluated on the full test dataset, the detection rate dropped to 53%-69%, and the classification accuracy decreased to 29%-42% as displayed in Figure 2.

This decrease in performance on the full test dataset correlates with the differences in data sources between the partial and full test sets. The partial test set comprised only simulated data and VCM-derived ephemerides, with no inclusion of TLEs. In contrast, the full test set included data from all three sources, with a significant proportion of TLE-derived ephemerides, which likely introduced greater variability. Moreover, the training data provided to participants was predominantly simulated, and none of the finalists reported resampling this data to mimic data sources with less frequent orbit updates. The reliance on unaugmented synthetic data may have contributed to the observed decrease in performance when confronted with the more diverse private evaluation dataset.

Table 3 and 4 report the final team rank and scores for **Phase I** and **Phase II**, and the average scores related to the quality of the technical write-ups, respectively. Regarding the criteria, clarity measures how well-written and easy to understand the report is; novelty evaluates the originality and innovativeness of the approach or methods used in the research; technical depth assesses the level of technical detail and explanation provided about the methodologies; reproducibility examines how easy it is for others to replicate the research, based on the clarity of instructions, availability of code, and parameter settings; insights value the conclusions drawn from the research in terms of meaningful interpretations of the findings.

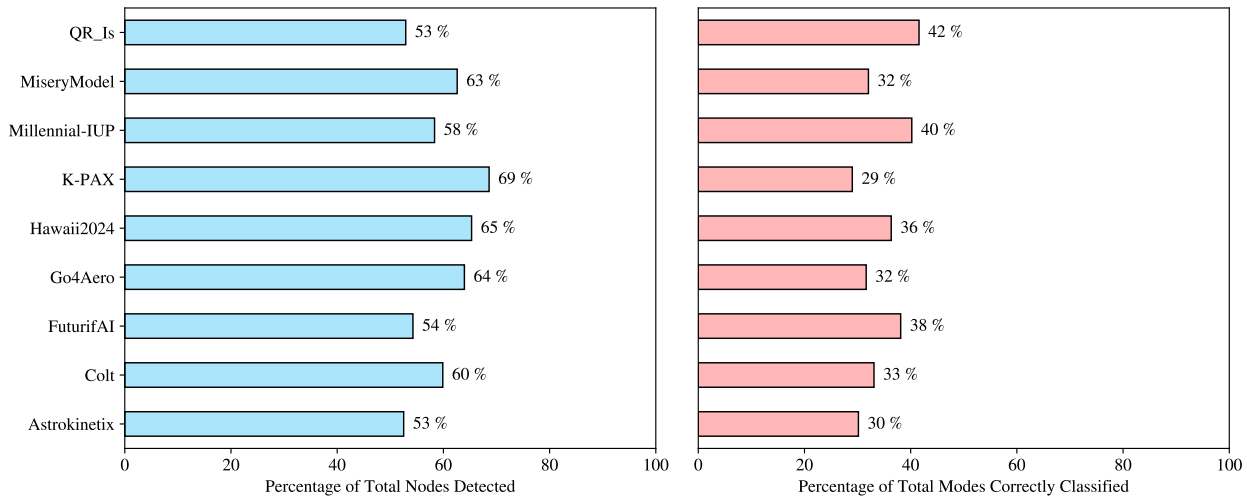


Fig. 2: Percentage of PoL nodes detected within 12 hours of truth (left) and station-keeping behavioral modes correctly classified (right) by each finalist team from the full evaluation dataset.

The following is a summary of the team-specific feedback from the judges:

- **Hawaii2024**: clear and concise writing, well-reasoned decisions, excellent reproducibility, and valuable insights. One judge questioned the validation set creation.
- **Millennial-IUP**: opinions were divided. Some lauded the code optimization, physics-informed approach, and quick development time. Others criticized a lack of focus, over-reliance on trial-and-error, and insufficient discussion of challenging aspects.
- **QR_Is**: highly regarded for the clear report, strong reproducibility, balanced approach, and data analysis leading to useful insights. One judge noted a disconnect between sections and questioned the practical application of some analyses.
- **MiseryModel**: received consistently positive feedback for exceptional clarity, meticulous documentation, clever pre-processing, and valuable insights. One judge noted difficulties reproducing CNN results due to variability in CNN training.
- **K-PAX**: novel approach with a focus on human factors and a custom loss function, but some judges found the report sparse on details and overly reliant on jargon. Questions arose about specific procedures.
- **Go4Aero**: appreciated for the sensitivity analysis, use of equinoctial elements, and attempts to address data disparity. One judge noted a lack of detail about model hyperparameters.
- **FuturifAI**: interesting comparison of models and important features, but missing detailed justifications and depth.
- **Astrokinetix**: praised for the understanding of astrodynamics and detailed explanation of heuristics. However, the lack of AI/ML was seen as a weakness.
- **Colt**: concise report which demonstrates promising insights about the challenges to handle high- and low-frequency data, but lacking technical depth and context.

Some similarities in the judges' feedback between the teams can be also found:

- **Hawaii2024**, **QR_Is**, and **MiseryModel** received high marks for clarity, reproducibility, and insightful analysis.
- **Millennial-IUP**, **QR_Is**, and **Astrokinetix** were commended for using domain knowledge (i.e., astrodynamics) to inform their models.
- **Go4Aero**'s sensitivity study was well-received, suggesting that judges appreciate attempts to assess robustness and understand model behavior under different conditions.

Whereas the following unique points were made:

- **Millennial-IUP**'s focus on code optimization was a unique highlight, although its effectiveness was debated.
- **K-PAX** was the only team to explicitly consider human factors in their pre-processing, which one judge found "powerful."

- **Astrokinetix**'s reliance on heuristics, while well-explained, was seen as a missed opportunity to explore ML's potential, highlighting an ongoing debate in the field.

The overall takeaways can be summarized as follows. (1) Clarity, reproducibility, and insight are crucial factors which were consistently valued by all judges, demonstrating their importance in communicating research effectively. (2) While innovation is important, judges had different perspectives on what constituted a sufficiently novel approach. (3) The domain expertise is valuable but needs to be coupled with ML.

Table 3: Final team rankings and scores for **Phase I**, **Phase II**, and the overall competition.

Rank	**Team**	**Phase I Score** (F_2)	**Phase I Score** ($F_{2,\text{norm}}$)	**Phase II Score** (Q)	**Final Score** (CS)
1	Hawaii2024	0.805	0.994	0.827	0.960
2	Millennial-IUP	0.810	1.000	0.713	0.943
3	QR_Is	0.793	0.979	0.787	0.941
4	MiseryModel	0.800	0.987	0.753	0.940
5	K-PAX	0.771	0.951	0.653	0.892
6	Go4Aero	0.771	0.952	0.640	0.890
7	FuturifAI	0.780	0.963	0.520	0.874
8	Astrokinetix	0.709	0.875	0.627	0.826
9	Colt	0.757	0.935	0.293	0.807

Table 4: Scores of technical write-ups by evaluation criteria.

Team	**Clarity**	**Novelty**	**Technical Depth**	**Reproducibility**	**Insights**	**Report Score**
Hawaii2024	3.833	3.667	4.500	4.500	4.167	20.667
QR_Is	4.167	3.667	4.167	4.000	3.667	19.668
MiseryModel	4.000	3.167	4.000	4.000	3.667	18.834
Millennial-IUP	4.000	3.167	3.833	3.500	3.333	17.833
K-PAX	3.333	3.333	3.667	2.333	3.667	16.333
Go4Aero	3.667	3.000	3.333	2.333	3.667	16.000
Astrokinetix	3.333	2.333	3.667	3.667	2.667	15.667
FuturifAI	3.000	2.333	2.000	2.667	3.000	13.000
Colt	2.000	1.333	1.333	0.667	2.000	7.333

3.1 Team Hawaii2024

As part of the MIT ARCLab Prize for AI Innovation in Space 2024, team Hawaii2024 developed a state-of-the-art satellite pattern-of-life characterization algorithm, achieving F2 scores of 0.95 and 0.81 in **Phase I** and **II** of the competition. The solution is highly customizable, incorporating custom CNN and LSTM architectures, tailored training methods, and extensive feature engineering, which may contribute to the development of future AI algorithms for SSA. The code is publicly available at GitHub[‡].

3.1.1 Algorithm Overview

The algorithm follows a three-step procedure to localize and classify PoL-changepoints of GEO satellites:

1. Preprocessing using feature engineering, dataset scaling and temporal windowing.
2. Temporal localization of PoL-changepoints using a sliding window approach and convolutional-dense neural networks. Unlike traditional binary labeling, which assigns a 0 (no changepoint) or 1 (changepoint) to each time index, this method employs a continuous labeling system that assigns values from 0 to 100. At actual changepoints, labels are set to 100, decreasing exponentially with distance from these points. This "gradient-encoding" teaches the model about the proximity of each time index to the nearest changepoint. Furthermore, it effectively reduces class imbalance and mitigates the issue of over-penalizing near-miss predictions.

[‡] https://github.com/DavidBaldsiefen/splid-challenge

3. Classification of the detected changepoints using a LSTM-based neural network. After the classifier predicts the most likely type-label (No or Chemical/Electrical/Hybrid Station-Keeping) for each changepoint, a set of logical rules assigns the corresponding class-label in the SPLID-format and removes unlikely outliers.

3.1.2 Preprocessing and Feature Engineering

A large number of feature combinations, including different transformations, have been evaluated during extensive hyperparameter studies using the Weights&Biases platform [5]. This process was further aided by the SHAP framework, which provides a game theoretic approach to explaining model outputs based on the inputs [6]. The Cartesian state features of the satellites were discarded early on, as they are a redundant representation of the more interpretable Keplerian elements [7].

The following feature transformations are performed in order to increase the expressiveness of the data:

- Cyclical Encoding: As most features show clear periodicity, a sine and/or cosine transformation may be applied. This aids in the representation of periodicities, while also removing discontinuities. [8]
- Differential Encoding: This transformation encodes a feature's gradient by computing the difference between consecutive values. It shifts the focus to the temporal dynamics within the data, emphasizing large local anomalies while also highlighting periodic changes as they would be visible in station-keeping behavior. Angle discontinuities are wrapped appropriately. This encoding often results in significantly improved performance over the sine/cosine transformation.
- Time Encoding: This feature encodes the current window's position in the object's time series using a one-hot representation, where an array of "0" values includes a single "1" to indicate the specific location of the current window relative to the entire series.

In addition to these transformations, every input feature is standardized by removing the mean and scaling to unit variance. The mean and variance are determined based on a combination of all available satellite time series. A low-pass filtering transformation was also applied to some of the input features with the goal of removing 24h oscillations, but showed no additional benefit.

A novelty of the proposed solution is that in addition to the input features, a transformation is also applied to the labels for the temporal PoL-changepoint localizers. This involves assigning a value of 100 to the label of each PoL-changepoint, with surrounding labels being assigned exponentially decreasing values at a rate of 0.7, stopping at a value of 16. Every other label is set to 0. This essentially encodes the proximity of each time step to the nearest changepoint, thereby reducing class imbalance. The labels for the classifier are also discretized using scikit-learn's LabelEncoder [9].

3.1.3 Windowing

All of the machine learning models used in this solution work using a windowing approach. Therefore, they make predictions for every time step based on previous, current, and future values of the original and transformed features. While larger windows may improve a model's 'overview' of the general dynamics in the data, they also increase computational costs and might lead to a large amount of overlap (and this similarity) between consecutive time steps. On the other hand, smaller windows offer deeper insights into local dynamics, potentially lacking the ability to encode long-term changes in a satellite's behavior. Furthermore, striding may be applied to each time window, where only data from every n-th time step is maintained. This transformation is performed similar to an average-pooling operation.

The optimal size and stride of the windows depend largely on the task at hand. For the temporal PoL-changepoint localizers, this mainly depends on the type of node to be predicted. While AD/IK nodes can be associated with propulsion maneuvers that leave a clear trace in the orbit ephemeris, ID nodes are usually characterized by changes in the long-term station-keeping behavior of a satellite. Therefore, AD/IK nodes are best localized using comparatively small windows at a higher temporal resolution (128-32, stride 2), while ID nodes are best identified using large windows which capture a broader horizon (320-256, stride 4). This motivates the decision to train individual machine learning models for each of these cases. To provide the ID-model with additional context of the long term changes in a satellites trajectory, it is further provided with a so-called overview feature. This contains a representation of the running standard deviation of the inclination of the RSO, downsampled to the size of the evaluation window.

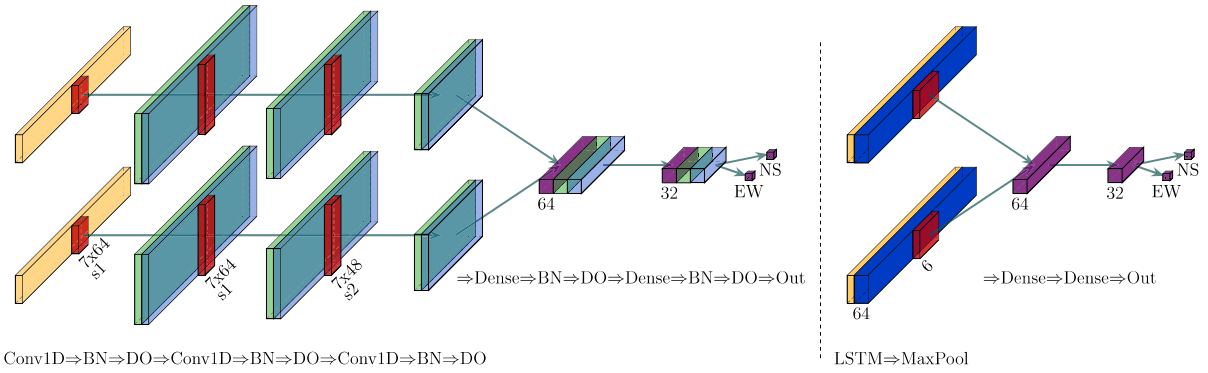


Fig. 3: Model architectures of team Hawaii2024. The localizers (left) consist of three 1D convolutions per input direction, followed by two shared dense layers. In addition, each layer is followed by one batch normalization and one dropout layer. The classifier (right) consists of one LSTM layer and one pooling layer per input direction, followed by two shared dense layers. It does not contain batch normalization or dropout layers.

Regular station-keeping maneuvers stand out in this representation as distinct periodic peaks, which can be easily captured by convolutional networks. This aids the model in understanding long-term changes in the station-keeping behavior, especially in the transverse (NS) direction. To contextualize this information, a one-hot encoding of the current window's position in the object's time series is also included.

While a PoL-Node's *timing* is largely driven by the satellites behavior during the time indices leading up to the changepoint, the *classification* of that Node is largely dependent on its behavior following the changepoint. This is reflected in the choice for the horizons of the classifier, which are very asymmetric at 32-256. The input resolution for the classifier is very high at a stride of 1; this is chosen mainly as it maximized accuracy on the high-frequency satellite ephemeris available during the competition phase. However, it is important to note that a larger stride increases the model's robustness towards interpolated data from satellites with low-frequency orbit updates, as it effectively suppresses high-frequency dynamics. The classifier is furthermore provided with overview features of the running standard deviation the satellite's argument of periapsis, latitude and longitude.

3.1.4 Model Architectures

As mentioned before, a total of three models are employed in the proposed solution. This includes the temporal PoL-changepoint localizers (one for AD/IK and one for ID nodes), as well as the changepoint classifier.

During the development of the localizers, it became apparent early on that detecting nodes in the transverse (NS) direction is significantly more challenging than the longitudinal (EW) direction. However, training individual models for each direction only gave marginal improvements, while effectively doubling model training times. Furthermore, many nodes are characterized by patterns which are shared across both directions (e.g. inclination changes can indicate maneuvers in both EW and NS direction), indicating that a unified model might be beneficial for detecting nodes in both directions. To overcome this challenge, the final architectures combine elements from both the unified as well as the individual approach. As such, each model's inputs are first fed into two identical branches, each consisting of three 1D convolutional layers. These are well suited for detecting localized patterns (such as propulsion maneuvers) in the data. Then, both branches are concatenated and the combined output is fed into two dense layers. Finally, two single-unit dense layers serve as output, representing each direction. To help the model generalize to new data, each dense and convolutional layer is further followed by a batch normalization and a dropout layer. Additionally, L2 regularization helps prevent overfitting by penalizing large weights in the model, encouraging simpler models with smaller weight magnitudes. While RNN based models have also been evaluated, it was not possible to perform sufficiently thorough parameter studies on such architectures for the localizers, due to the author's limited computing resources.

The final dual-branch architecture has several advantages: it offers reduced training times, while still allowing the model to learn distinct as well as shared patterns relevant for each direction. While it could be argued that the same

should be achievable by simply doubling the number of filters in each convolutional layer, experimental results showed a clear benefit from splitting up the layers. Several other configurations, such as splitting up the dense instead of convolutional layers, have been attempted as well, but showed no additional advantages. A visualization of the final architecture can be found in Fig. 3.

Similar to the architecture of the localizers, the input of the classifier also feeds into two identical branches, followed by two dense layers. However, instead of using a sequence of convolutional layers, each branch consist of a single LSTM layer followed by a max-pooling layer. The idea to use LSTMs instead of CNNs stems from the insight that for the classification, we are not interested in a distinct, local state change. Instead, the classification of any of the timesteps following the node should be identical (i.e. station-keeping or not station-keeping). Therefore, a LSTM that is configured to return its hidden state for every timestep in the sequence should return a similar state for every timestep. This is further exploited by the pooling layer, which can then downsample the output of the LSTM without a significant loss of information. The final classifier model reaches an accuracy of over 95% on the **Phase I** test set, underlining the effectiveness of this approach. Notably, it is configured without any dropout or batch normalization layers. This suggests a high robustness of the extracted features, and hints towards the possibility of further simplifying the model, enhancing its ability to generalize to unseen data.

While these architectures did lead to the best overall results of the teams proposed solution, it is important to note that the model's exact configurations had far less impact on the overall performance than the definition of the inputs (features, window size, preprocessing), underlining the importance of rigorous feature engineering.

3.1.5 Training and Postprocessing

Each model was trained using 80% of the available data, with 20% being held out for validation purposes. Only the final localizers were trained on a 95/5 split, in order to maximize the utilization of available data samples. While the localizers were trained for a fixed number of 90 epochs, the classifier was trained for up to 500 epochs with early-stopping if performance on the validation set did not improve for 70 epochs.

To further address the significant class imbalance between changepoints and non-changepoints within the dataset, a specialized striding technique is implemented within the sliding window approach during training. Rather than applying the window at every consecutive time index, we selectively sample every n-th time index, optimizing computational efficiency. Notably, time indices indicating changepoints (label > 0) are always included, ensuring critical points are thoroughly analyzed regardless of striding. This synergizes well with the fact that the time indices immediately surrounding a changepoint are also assigned labels > 0. This strategic approach effectively resamples the dataset, emphasizing changepoints while minimizing computational overhead in regions of temporal stability. To further maximize use of the available data, training halts after a set number of epochs, followed by resampling the dataset at an offset of $n/2$. For instance, if the initial training covers time indices [0,4,8,...], subsequent training epochs focus on indices [2,6,10,...], ensuring comprehensive model training across the entire time series.

Even though the classifier is trained using the same general procedure, the dataset-stride is set very high at $n = 1000$. This means that the model is trained almost exclusively on actual changepoints, with only a small amount of data samples stemming from other areas in the objects' time series. The advantage of this method is that the model is trained in conditions much more similar to those it will see during inference, where it is only tasked with classifying time indices that have already been determined to be changepoints.

In order to turn the raw outputs of the models into a usable data structure, a couple of postprocessing steps have to be performed. First, the linear output of the localizers is binarized again, by applying a simple threshold to each output. These binary labels are then grouped, so that consectuive "True" predictions are replaced by a single changepoint prediction at their center. Changepoints predicted by the ID-localizer are classified as ID automatically, while the remaining changepoints are associated with the type-label predicted by the classifier at that node. The corresponding class-labels can then be determined using a simple set of logical rules. For example, if a node is classified as station-keeping with the previous node being classified as not station-keeping, the current node's class has to be "Initiate Station-keeping". As NS nodes can only occur during EW station-keeping behavior according to the challenge definition, NS nodes outside of this boundary are automatically removed. This made sense in the context of the competition as EW predictions are consistently more accurate than those in NS direction.

A distinct advantage of binarizing the linear output of the localizers using custom thresholds is the ability to fine-tune

the trade-off between precision and recall according to the user's needs. While this study focused on optimizing the F_2 score, lower thresholds can be applied to increase the model's recall, while larger thresholds increase the model's precision. This flexibility suggests the development of adaptive thresholding algorithms, which could enhance AI-driven satellite monitoring systems by dynamically adjusting decision boundaries based on real-time data.

3.1.6 Summary and Insights

This section introduced a state-of-the-art satellite pattern-of-life characterization algorithm. The proposed method effectively mitigates class imbalance issues inherent in traditional changepoint detection algorithms by providing a continuous gradient representation of PoL-changepoints, and utilizing efficient dataset resampling techniques. Splitting the task over several models with varying input horizons is an effective way to exploit the distinct patterns of different maneuver types. Furthermore, detection and classification performance in both longitudinal and transverse direction is enhanced by using a custom dual-branch architecture. This is complemented by extensive feature engineering, including differential encodings and overview features, which may also help enhance future AI SSA algorithms.

Finally, the proposed method is highly customizable, allowing easy adjustments to balance recall and precision, potentially offering a promising solution for a wide range of changepoint detection applications beyond the satellite domain.

3.2 Team Millennial-IUP

After discovering the MIT ArcLab AI Innovation in Space competition only 3 weeks before the Phase I submission deadline, the team moved fast. Millennial Software Solutions Inc. and Indiana University of Pennsylvania (IUP) committed to working together to implement an innovative solution to characterize GEO satellites PoL using cutting edge ML techniques. Together, the team executed a fast-paced research and development effort to successfully develop a ML capability for the competition, leveraging collaboration amongst industry veterans and student researchers.

3.2.1 Code Optimization

After performing an initial code inspection and trial run on the codebase from the MIT ARCLab splid-devkit repository, the team realized that the codebase needed to be tailored to fit their needs. Several optimizations were implemented that allowed for more experimentation within the tight schedule than would otherwise have been possible. Updates included parallelization, efficiency of concatenation, and usage of graphical processing unit (GPU) compute resources.

3.2.2 Feature Engineering

After code optimization and adoption of the XGBoost model, the team incorporated intuitive expert knowledge in deriving new features. The devkit provided lag columns, and the team built on this with lead columns. Lead columns were like lag columns, except they held values from future timesteps.

The lag columns were initialized with "Not a Number" (NaN) values near time t=0 (TimeIndex 0). By default, the devkit backfilled data by replacing NaN values with the next non-NaN value in time. The team removed this backfill copying which greatly aided prediction of SS Node labels. Since XGBoost accepted NaN input values, their presence in lag features near time t=0 gave the model a definitive deciding value for properly labeling SS Nodes.

Due to the nature of the GEO orbit having an approximately 24-hour orbital period, incremental parameter search was performed between 2 to 48 hours windows (half lag and half lead) around each data point. Based on human expert knowledge, the theory was that by giving the models up to 2 full orbital periods worth of input, there would be enough temporal context to enable pattern recognition. To optimize the context, application of parameter search over the orbital period was conducted independently for each ephemeris feature. Varying arrays of context permutations were used for the various submissions via feature schedules. The most performant model, however, made use of a common context of 24 hours for all features. This makes sense intuitively and aligns with the original context thesis.

Satellite operators initiate longitudinal (EW) drift maneuvers by reducing or increasing altitude. A significant change in altitude is correlated with a transition from station-keeping to drifting in either direction. For the competition,

```
feature_schedule = [
          ("Eccentricity",               [-12,-6,-1,0,1,6,12]),
          ("Semimajor Axis (m)",         [-12,-6,-1,0,1,6,12], "SMA Delta (m)"),
          ("Inclination (deg)",          [-12,-6,-1,0,1,6,12], "Inclination Delta (deg)"),
          ("RAAN (deg)",                 [-12,-6,-1,0,1,6,12]),
          ("Argument of Periapsis (deg)",[-12,-6,-1,0,1,6,12]),
          ("True Anomaly (deg)",         [-12,-6,-1,0,1,6,12]),
          ("Latitude (deg)",             [-12,-6,-1,0,1,6,12]),
          ("Longitude (deg)",            [-12,-6,-1,0,1,6,12]),
          ("Altitude (m)",               [-12,-6,-1,0,1,6,12]),
          ("Z (m)",                      [-12,-6,-1,0,1,6,12]),
          ("Vz (m/s)",                   [-12,-6,-1,0,1,6,12]),
]
```

Fig. 4: Feature Schedule 2.1.0.0: The features X, Y, Vx, and Vy were intentionally omitted from the list of available features, thereby excluding them from the model. The offset list [-12,-6,-1,0,1,6,12] defines three lag columns at 12, 6, and 1 timesteps in the past, along with three lead columns at 1, 6, and 12 timesteps in the future. For the Semimajor Axis and Inclination, delta columns were also specified, with each lag/lead column having a corresponding delta column at the same offset.

the team chose the closely related semimajor axis column and derived a new set of columns. Delta features were defined to be the difference between the current timestep and its corresponding lag (or lead) column. To represent changes to orbital attributes, delta values were considered for all orbital elements: Eccentricity, Semimajor Axis, Inclination, RAAN, Argument of Periapsis, and True Anomaly. Based on expert knowledge, the team determined that the semimajor axis (delta_SMA) and inclination (delta_INC) were the two effective attributes of changes, which were adopted for the final feature vector. Upon creation of each lag/lead column for semimajor axis, the corresponding SMA Delta lag/lead was also created. The same process was conducted for inclination. Any large absolute value in the delta attribute contributed a significant signal of a state transition. An outlier value coupled with a transition label created an easy-to-set threshold boundary for prediction. The inclusion of delta_SMA and delta_INC features in the model accounted for a 0.0091 (out of 1.0000) F_2 score increase on the local validation set. The result of this experiment aligned with expert knowledge since delta_SMA is a direct result of an in-track maneuver and delta_INC is affected by a cross-track maneuver.

The team created feature schedules to provide instructions for the updated preprocessing code that specified the columns to be used for loading, training, and prediction. The team heavily modified the preprocessing to populate temporal columns as instructed by each feature schedule. These updates gave the team a flexible and precise method for deriving temporal features and allowed the specific offsets of lag/lead to vary from feature to feature. Moreover, XGBoost is sensitive to input column order, and feature schedules let the team experiment with different permutations within the input DataFrame. Domain knowledge guided the creation of an array of feature schedules, with small variations of the columns between them. They formed a tree hierarchy and were numbered with a system akin to software versioning. Numbering the schedules helped immensely in results tracking and communication. The team's highest scoring submission was trained with feature schedule 2.1.0.0, shown in Fig. 4.

3.2.3 Automated Parameter Search

Feature schedules provided a means to experiment with a variety of permutations at different offsets. However, evaluating each schedule required manually altering inputs, rerunning the notebook, and collecting the results. With code optimization, each notebook execution duration was reduced from approximately 50 minutes to between 5-8 minutes. Although possible, manual execution of the number of permutations of parameter configurations was impractical and prone to human error. Therefore, the team sought to script the process of running the notebook. The papermill project provided a way to parameterize and execute Jupyter Notebooks, enabling the use of a driver application architecture to execute the ml_baseline notebook in a separate Python script. This script executed all the planned feature schedules and aggregated model performance metrics over the runs for evaluation, which is visualized in Fig. 5. The team used papermill to efficiently and consistently experiment in the search for the most performant schedule. This innovative approach to parameter search automation significantly reduced time, error, and enabled tuning for the training process.

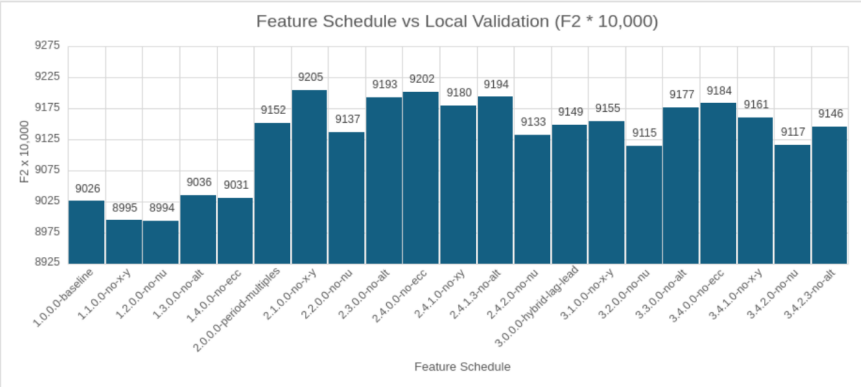


Fig. 5: Aggregate results after 21 notebook runs using papermill. The F_2 score was scaled up by 10,000 to eliminate decimal points for clarity. Each schedule was categorized into one of three groups, each originating from an initial ancestor numbered as x.0.0.0. Among these, 2.1.0.0 exhibited the highest F_2 score.

3.2.4 Transition Labels

The dataset included intermittent *transition labels* indicating shifts between drifting (ID/AD) and station-keeping (IK) states. Rather than using the devkit's preprocessing to generate *state labels*, we opted to train XGBoost directly on the original transition labels. The SMA Delta columns often had outlier (large magnitude) values in the vicinity of timesteps of EW drift maneuvers (corresponding to e.g. label IK-EK). SMA Delta values outside the vicinity were not outliers; they were closer to mean 0. With transition labels, IK-EK only appeared on the transition timestep, and the outlier-to-label correlation was clearly marked for learning by XGBoost. By contrast, with state labels, IK-EK was also applied to many timesteps with SMA Delta values close to 0 (non-outliers). By labeling IK-EK onto numerous timesteps without outlier values, the outlier-to-label correlation disappeared. The switch to transition labels was motivated by the concept of preserving the outlier-to-label correlations. This same concept also applied more generally to other feature values that correlate with transition events, but SMA Delta was the clearest example, as seen in Table 5.

Table 5: Excerpt of Object 285's SMA Delta (m)_lag_1 from $t = 1413$ to $t = 1450$, highlighting the transition to station-keeping at TimeIndex 1415 (grey). The outlier value at 1415 clearly indicated the transition in the transition labels but did not correlate with the state labels.

TimeIndex	SMA Delta (m)_lag_1	Transition Labels	State Labels
1413	21.85	NaN	AD-NK
1414	802.80	NaN	AD-NK
1415	-133684.17	IK-EK	IK-EK
1416	62.15	NaN	IK-EK
...	...	...	...
1450	450.34	NaN	IK-EK

Other ML models may perform better with state labels, but XGBoost gained F_2 in excess of 0.70 (out of 1.00) after switching to use transition labels exclusively throughout the process. The number of False Positives (FPs) predicted on the local validation set dropped from 17,950 to 50 in the switch from state labels to transition labels. This change was by far the largest contributor to the model's performance. Figure 6 illustrates the difference while focusing on one of the objects from the validation set.

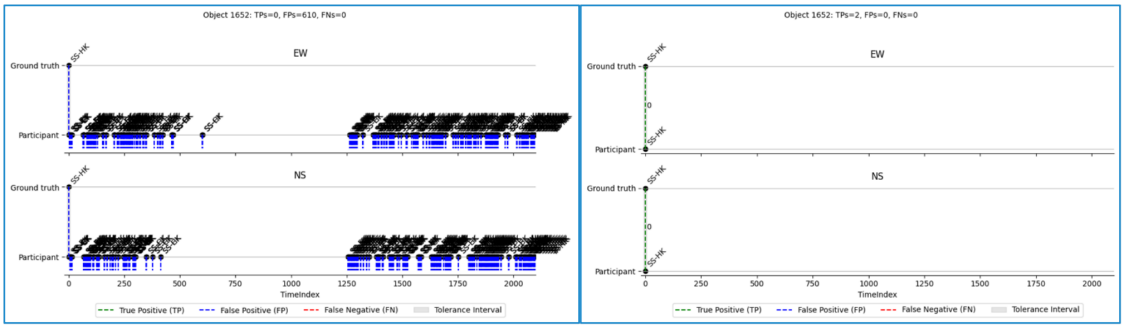


Fig. 6: Predictions for object 1652. Left Plot: Using state labels, 610 false positives (FPs) are shown as blue lines. Right Plot: Using transition labels, there are 0 FPs. The ground truth for this object consists of only two SS-HK labels, one for each of the NS and EW directions. Plots were generated using the devkit.

3.2.5 XGBoost Hyperparameter Tuning

Upon achieving substantial performance with updated data preprocessing methods, the team evaluated hyperparameter tuning to improve upon the default XGBoost models. XGBoost has a number of booster hyperparameters that determine how trees are constructed. Furthermore, SKLearn's RandomizedSearchCV allowed for quickly searching through hyperparameter combinations to find the optimal hyperparameters for the model and dataset. The following XGBoost hyperparameters were tested: max_depth, learning_rate, and subsample. The max_depth represents the maximum tree depth for base learners, learning_rate affects how much successive trees differ, and subsample is the ratio of the training instance. The optimal hyperparameter combination was found to include max_depth=8, learning_rate=0.3, and subsample=0.7, and together they contributed 0.0068 (out of 1.0000) in F_2 score against the local validation set.

3.2.6 Summary and Insights

The Millennial-IUP team successfully leveraged the MIT challenge dataset to train a ML model to effectively predict GEO space object maneuver events. The end-to-end training process encompassed modification of the ML baseline code, feature engineering, expert knowledge encoding, hyperparameter tuning, and automated parameter search. Leveraging subject matter expertise and experience in the SDA field, the team was able to encode GEO orbit regime astrodynamics characteristics into effective features. Experiment parameterization and automation allowed the team to rapidly search through the parameter space for optimal feature vectors. The knowledge gained from this competition can inform the process for future SDA applications of maneuver detections. The national security SDA problem space often involves maneuver detection as the first step in the processing chain. Therefore, the lessons learned from this competition will drive innovative solutions for national defense and beyond.

3.3 Team QR_Is

Representing Team QR_IS, we achieved fourth place in Phase I of the competition. Despite not topping the leaderboard, our model demonstrated the best balance in performance, extrapolation ability, and computational efficiency. Our results underscore our model's robustness, simplicity, and effective resource management, making it ideally suited for practical deployment in space traffic management.

3.3.1 Feature Engineering and Selection

A feature is a transformation of the raw data that is fed as input to a model. In this section, we detail our highest quality features with important predictive power. You can find the exhaustive list of features on our GitHub[§].

Trajectory-Based Features (257 features) Our analysis uses 257 features from trajectory analysis, derived from various time-series transformations to detect changes and patterns. These include Lag *(180 features)*, Difference

[§] https://github.com/FrancoisPorcher/mit-challenge

(36 features), Percentage Change *(6 features)*, Rolling Average *(10 features)*, and Rolling Standard Deviation *(10 features)*. For example, the percentage change in eccentricity is shown in Fig. 7, highlighting its activity near a node. Furthermore, the feature termed "envelope", which represents the rolling minimum and maximum of a time-series, demonstrates significant predictive power. More specifically, fluctuations in the distance between the maximum and minimum values of the envelope often coincide with the occurrence of a node, as illustrated in Fig. 8, where the feature becomes active at the node.

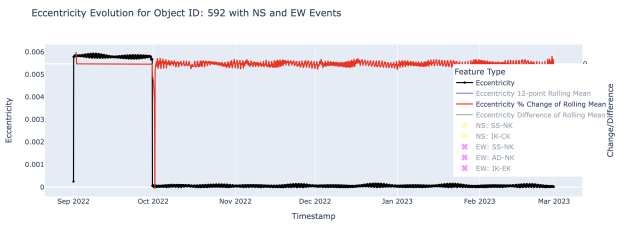


Fig. 7: Eccentricity change.

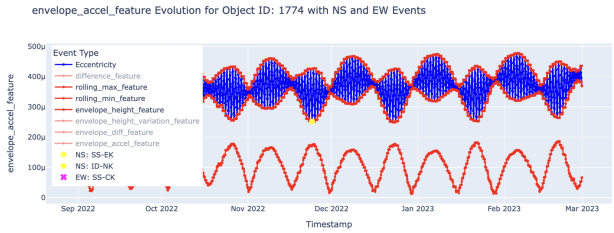


Fig. 8: Envelope difference.

Satellite Characterized Features (16 features) In this section, we explore features that consistently characterize satellites. Notably, we observe that satellites with high variance in their features often have a greater number of nodes. This relationship is demonstrated in Fig. 9, which shows a correlation between the total standard deviation of eccentricity and the number of nodes in the East-West (EW) direction. This finding is particularly useful for our predictive model, as it can leverage knowledge of high volatility at any time step to trigger a prediction more effectively. The specific features derived include average values across the trajectory *(9 features)* and standard deviation *(7 features)*.

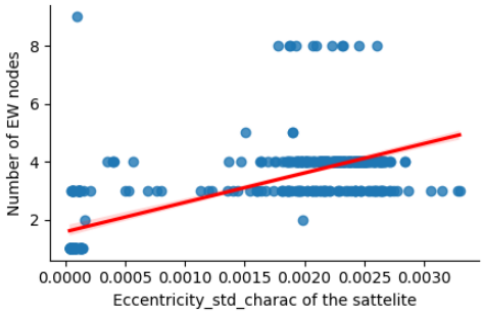


Fig. 9: Scatter plot for 500 satellites (each point represents one satellite).

Time-Based Features (5 features) We developed five time-based features that capture cyclical patterns to reflect temporal progress in our model. For example, one feature cycles through values from 1 to 12 over 12 timestamps, each representing two hours, equating to one day. These features, including cycles of 12, 24 (one day), 72 (three days), 168 (one week), and 336 hours (two weeks), allow the model to recognize time-related patterns, such as node occurrences more likely towards the end of a day. This concept is akin to positional encoding in transformers [10], which tracks sequence positions.

Insights on Frequency Analysis During the initial phase of the challenge, we hypothesized that the periodic nature and noticeable seasonality in the data suggested that frequency patterns possess significant predictive power. As demonstrated in Fig. 10, there is a noticeable change in frequency within the Y (m) data around nodes. Specifically, the envelope exhibits high frequency fluctuations before a node and slows down afterwards. This is confirmed by a Fast Fourier Transform (FFT) analyses of the time series data. After the node, the FFT reveals that the magnitudes increased in the zone of longer periods, suggesting a slower modulation of the envelope [11]. To quantify these observations, we project the time series data from a three-day rolling window onto a Fourier basis and compute the *L2* distance between successive vectors. This approach has proven effective in capturing the peak frequency changes during events, as shown in Fig. 11.

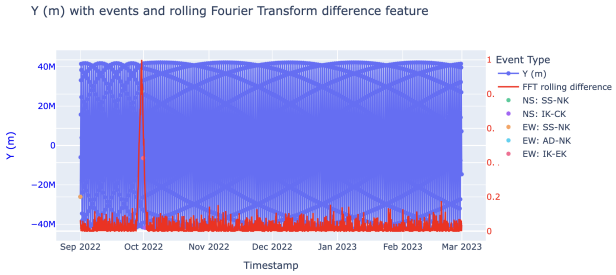


Fig. 10: Change in envelope frequency before and after the node.

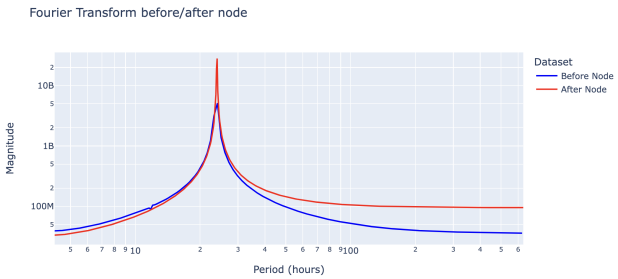


Fig. 11: FFT Transform before and after node.

3.3.2 Methodology

Problem formulation Each timestamp records the *node* for North-South (NS) and East-West (EW) directions. The goal is to identify *nodes* within a 6 time-steps tolerance, treating it as two separate multi-class classification tasks, one for each direction. Although this could be treated as a single multi-class classification task, the possibility of simultaneous events in both directions necessitates handling them as separate problems. This separation is essential as a single classifier per timestamp might overlook a concurrent event. Further justification is provided in the Model Separation and Recombination Strategy section.

Model choice We selected gradient-boosted trees as our primary model, using the CatBoost Python package. This decision was driven by the small training dataset, which limits the effectiveness of data-intensive architectures like deep neural networks or transformers [12]. Given these dataset constraints, our choices were narrowed to gradient-boosted trees and shallow neural networks. We opted for gradient-boosted trees due to several key advantages: they are faster to train, less prone to overfitting, resources efficient, robust against outliers, and explainable. Additionally, they had tremendous success in the competitive data science community [13, 14].

Table 6: Comparison of Gradient-Boosted Trees and Neural Networks in Multi-class Classification.

Feature	Gradient-Boosted Trees	Neural Networks
Complexity	Easy, only a few hyperparameters to tune.	High. Requires architecture choices and tuning.
Training	Fast training, low ressources, less prone to overfitting.	Slow training, requires expensive GPU, more prone to over-fitting, training less stable.
Interpretability	High, with clear decision paths.	Low, considered a "black box."
Noise Handling	Robust against data noise and outliers.	Sensitive; often requires additional steps.

Model Separation and Recombination strategy This section evaluates the decision to use separate models for NS and EW directions, and describes our recombination strategy to share information between models. Here is the trade-off between using a simple model vs two independent models for each direction:

Table 7: Pros and Cons of Model Separation Strategies.

	Single Model	Two Independent Models
Pro	• Accounts for NS-EW interaction • Simpler development and monitoring	• More control over precision and recall per direction • Can predict two events simultaneously
Con	• Reduced control over individual direction metrics • Difficulty managing 16 classes simultaneously	• Assumes independence of EW and NS • Longer development due to individual tuning

Recombination Strategy Our approach integrates independent predictions with interactions between NS and EW

through a recombination mechanism. Model 1 uses N input features for initial NS predictions (then discarded). Model 2 then adds these class probabilities to N features to determine EW. Finally, Model 3 uses Model 2's class probabilities and N features to refine NS predictions.

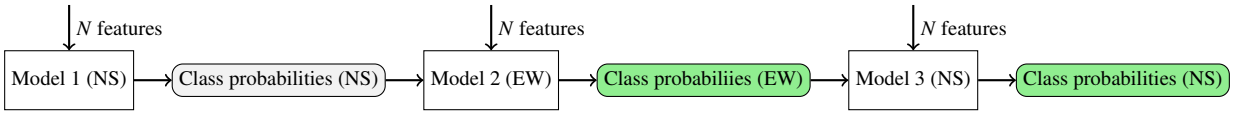


Fig. 12: Flowchart of the recombination strategy.

Model Stacking Model stacking enhances prediction accuracy by using multiple models sequentially. The heuristic model from challenge organizers tracks changes in satellite longitude and inclination, providing station-keeping and propulsion insights. This method is somewhat orthogonal to our tree-based models, adding diversity. We integrated this heuristic model into our pipeline, using its predictions as additional inputs for the tree-based model, as shown in the diagram below:

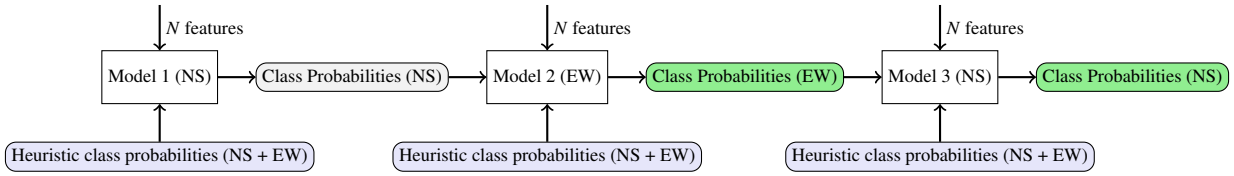


Fig. 13: Flowchart of the model stacking strategy.

Model Prediction Strategy This section describes our strategy to derive classes from model-generated probabilities. Notably, the F_2 score prioritizes Recall, weighting it four times more than Precision. This insight inspired a dual-threshold mechanism introducing two degrees of freedom to better manage precision and recall trade-offs in multi-class classification, vital for handling multiple thresholds in unbalanced datasets.

1. Highest Probability and Class $(p_{\max}, c_{\max})$: Represents the model's top prediction. If $p_{\max} \leq \tau_{\text{precision}}$, the prediction is set to "no event", ensuring that classes other than "no event" are sufficiently confident, thereby increasing precision.
2. Second Highest Probability and Class $(p_{\text{second max}}, c_{\text{second max}})$: Represents the second most probable class. If $c_{\max}$ is "no event" and $p_{\text{second max}}$ exceeds the recall threshold (τ_{recall}), the model opts for $c_{\text{second max}}$, thus prioritizing recall. This strategy is particularly useful in unbalanced datasets to avoid bias toward the most frequent class.

Parameter values are documented for reproducibility. Notably, the τ_{recall} for NS is higher than for EW to address the greater class count in EW (10) versus NS (9). This higher threshold helps improve recall, as more classes typically reduce the model's accuracy per class.

Table 8: Sensitivity Thresholds.

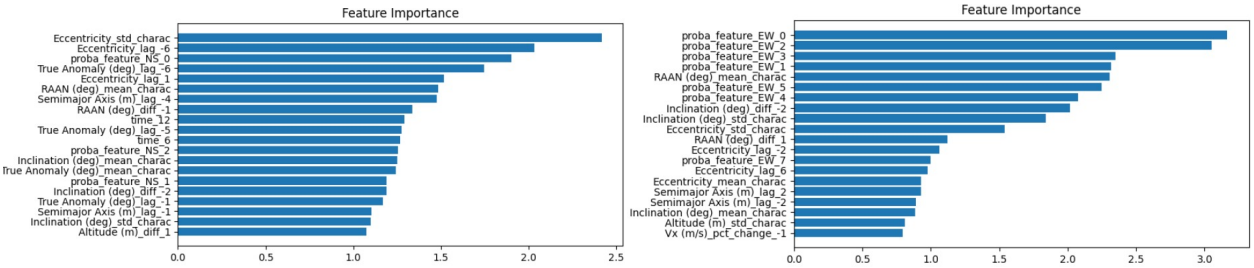
Direction	$\tau_{\text{precision}}$	τ_{recall}
EW	0.1	0.11
NS	0.1	0.26

3.3.3 Core Observations and Reproducibility

Features Importance The feature importance plots below validate our modeling choices by highlighting the predictive power of specific features, leveraging the high interpretability of decision trees [15].

The plots confirm that features related to Eccentricity and the outputs from previous models (proba_features) are critically important. This supports our strategies of model stacking and model recombination. Additionally, the significance of "Satellite Characterized Features" highlights the benefits of pre-informing the model about the satellite types it is analyzing.

Training Details, Hardware and Software Environment Initially, we trained using 500 trajectories and validated with 1,400. For the final model training, all 1,900 trajectories were utilized. Minimal hyper-parameter tuning was



(a) Model 2 (prediction of EW node).

(b) Model 3 (prediction of NS node).

Fig. 14: Top 20 feature importance for each model.

implemented, with the primary adjustment being an increase in *n_estimators* from 100 in experimental testing to 1,000 for final training to ensure model robustness without overfitting, which was confirmed by the final evaluation. Training was done on a CPU, demonstrating that deep learning's higher computational demands and cost are not always necessary. At equal computational resources, our model leads to superior performance. The full code and guidelines for replication are available here[¶].

3.3.4 Summary and Insights

Our analysis underscores the importance of satellite-specific features and envelope variations for prediction. Using frequency analysis, we established that seasonal and cyclical changes are key predictors. Techniques like Model Recombination and Model Stacking have refined our predictions, while a dual-threshold mechanism optimized the precision-recall trade-off. Although not leading the leaderboard, our model's simplicity, computational efficiency, and robustness make it suitable for real-world applications. For future enhancements, we recommend a deeper investigation in spectral analysis. With wavelet transforms, we anticipate more nuanced detections of cyclical patterns, potentially leading to significant performance gains.

4. CONCLUSION AND FUTURE WORK

The MIT ARCLab Prize for AI Innovation in Space 2024 challenge, focusing on the innovative application of artificial intelligence (AI) to satellite pattern-of-life identification, has demonstrated the significant potential of AI in enhancing space domain awareness (SDA). The diverse participation from global teams shows the widespread interest in both the AI and SDA community. The challenge results highlight the efficiency and robustness of AI models in automating the analysis of large satellite data, identifying behavioral patterns, and detecting anomalies with high precision.

Key takeaways from the challenge include the importance of data pre-processing and feature engineering. The competition showcased a broad spectrum of model approaches, from CNNs and LSTMs to random forests and gradient boosting algorithms. A common strategy among participants was the use of separate models for east-west and north-south PoL features, with many teams also incorporating statistical features and lagged inputs. Despite this diversity in methodologies, no single approach consistently dominated the leaderboard.

The performance of the finalist teams revealed some difficulty in generalizing from simulated to real-world data. While detection and classification rates were relatively high on the partial test dataset, which included only simulated and VCM-derived ephemerides, performance dropped notably on the full test dataset that included a significant proportion of TLE-derived ephemerides. This decline is likely linked to the greater variability and complexity of the full dataset, exacerbated by the lack of resampling in the predominantly simulated training data. These results underscore the challenges posed by the range of noise and sparsity represented by multiple, sometimes competing data sources, particularly when transitioning from simulated environments to real-world scenarios. Future efforts could benefit from data-augmentation techniques as well as more balanced incorporation of real-world data during the training phase and higher-fidelity data sources during the testing phase. These steps can potentially improve model robustness across different data sources, ultimately leading to better generalization and performance.

[¶] https://github.com/FrancoisPorcher/mit-challenge

ACKNOWLEDGEMENT

This research was sponsored by the Department of the Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. The authors would also like to thank the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC, database, and consultation resources that have contributed to the research results reported in this paper.

REFERENCES

[1] Peng Mun Siew, Haley E Solera, Thomas G Roberts, Daniel Jang, Victor Rodriguez-Fernandez, Jonathan P How, and Richard Linares. Ai ssa challenge problem: Satellite pattern-of-life characterization dataset and benchmark suite. In *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, page 5, 2023.

[2] Peng Mun Siew, Haley E Solera, Thomas G Roberts, Daniel Jang, Victor Rodriguez-Fernandez, Jonathan P How, and Richard Linares. Splid development kit. https://splid-devkit.readthedocs.io/en/latest/README.html, 2023. Accessed: 2024-07-21.

[3] Deshraj Yadav, Rishabh Jain, Harsh Agrawal, Prithvijit Chattopadhyay, Taranjeet Singh, Akash Jain, Shiv Baran Singh, Stefan Lee, and Dhruv Batra. Evalai: Towards better evaluation of ai agents.

[4] Juan Luis Cano Rodríguez, Yash Gondhalekar, Antonio Hidalgo, Shreyas Bapat, Nikita Astrakhantsev, Chatziargyriou Eleftheria, Kevin Charls, Meu, Dani, Abhishek Chaurasia, Alberto Lorenzo Márquez, Dhruv Sondhi, Tomek Mrugalski, Emily Selwood, Manuel López-Ibáñez, Orestis Ousoultzoglou, Pablo Rodríguez Robles, Greg Lindahl, Syed Osama Hussain, andrea carballo, Andrej Rode, Helge Eichhorn, Anish, sme, Himanshu Garg, Hrishikesh Goyal, Ian DesJardin, Matthew Feickert, and Ole Streicher. poliastro/poliastro: poliastro 0.17.0 (SciPy US '22 edition), July 2022.

[5] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[6] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[7] MA Hapgood. Space physics coordinate transformations: A user guide. *Planetary and Space Science*, 40(5):711–717, 1992.

[8] Tanisha Mahajan, Gaurav Singh, Glenn Bruns, G Bruns, T Mahajan, and G Singh. An experimental assessment of treatments for cyclical data. In *Proceedings of the 2021 Computer Science Conference for CSU Undergraduates, Virtual*, volume 6, page 22, 2021.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[11] John G. Proakis and Dimitris K. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. Pearson, 4 edition, 2006.

[12] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2021.

[13] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[14] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[15] Andrei V. Konstantinov and Lev V. Utkin. Interpretable machine learning with an ensemble of gradient boosting machines. *arXiv preprint arXiv:2101.03334*, 2021.