A SIMPLE MODEL AND ADVANCED STRATEGIES ARE ALL YOU NEED

Isaac Haik, Francois Porcher
isaac.haik@gmail.com

1 Introduction

Representing **Team QR_IS**, we achieved fourth place in Phase I of the competition. Despite not topping the leader-board, our model demonstrated the best balance in **performance**, **extrapolation ability**, and **computational efficiency**, as shown in Table 1. Our results underscore our model's robustness, simplicity, and effective resource management, making it ideally suited for practical deployment in space traffic management.

Table 1: Results of the competition

Rank	Participant team	F2 Total	F2 Partial	Difference
1	Millennial-IUP	0,811	0,917	-0,11
2	Hawaii2024	0,805	0,952	-0,15
3	MiseryModel	0,801	0,944	-0,14
4	QR_Is	0,795	0,852	**-0,06**
5	FuturifAI	0,781	0,822	**-0,04**
6	Go4Aero	0,773	0,916	-0,14
7	K-PAX	0,772	0,935	-0,16
8	Colt	0,758	0,949	-0,19
9	Yzzzz	0,731	0,864	-0,13
10	Astrokinetix	0,709	0,829	-0,12

2 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.

2.1 Trajectory-Based Features (257 features)

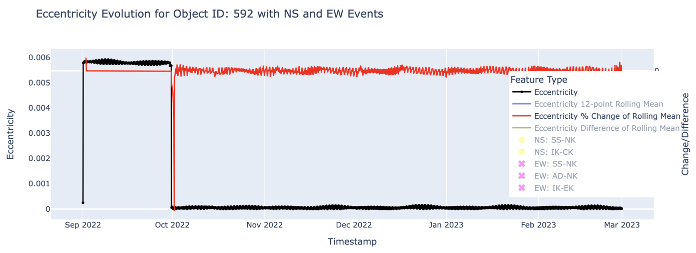


Figure 1: Eccentricity change

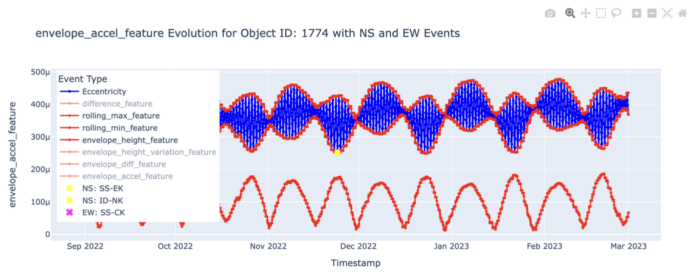


Figure 2: Envelope diff

A significant portion of our features, totaling 257, are based on trajectory analysis. These features are derived by applying various transformations to time-series data to identify changes and patterns. These transformations include Lag *(180 features)*, Difference *(36 features)*, Percentage Change *(6 features)*, Rolling Average *(10 features)*, and Rolling Standard Deviation *(10 features)*. For instance, as depicted in Figure 1, we calculate the percentage change for eccentricity and observe that the feature is particularly active in the presence of 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 Figure 2, where the feature becomes active at the node.

2.2 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 Figure 3, 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.

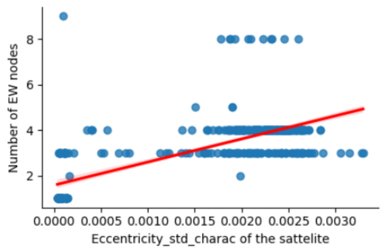


Figure 3: Scatter plot for 500 satellites (each point represents one satellite)

The specific features derived include average values across the trajectory *(9 features)* and standard deviation *(7 features)*

2.3 Time-Based Features (5 features)

We have developed a set of time-based features that introduce cyclical patterns reflecting the passage of time into our model. Each feature corresponds to a specific period and cycles through its range within that timeframe. For instance, one of our features cycles through values from 1 to 12 over a period of 12 timestamps, where each timestamp represents two hours. Thus, after n timestamps, $n/12$ days have elapsed.

This cyclical nature allows the model to incorporate temporal context in its predictions. For example, by analyzing a 12-timestamp cycle (equivalent to a 24-hour day), the model can learn patterns such as the increased likelihood of node occurrences towards the end of the day. The five time-based features we have implemented correspond to cycles of 12 hours, 24 hours (1 day), 72 hours (3 days), 168 hours (7 days), and 336 hours (14 days). These features help the model to be aware of the time, enhancing its ability to predict events based on temporal patterns. Similar features known as **positional encoding** are used in transformers, to keep track of the position in a sequence [1].

2.4 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 Figure 3, 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 [2].

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 Figure 4.

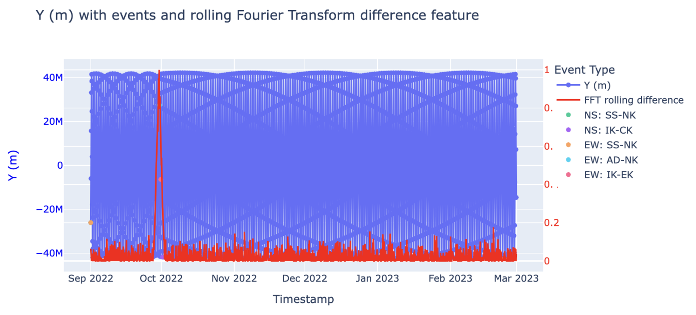


Figure 4: Change in envelope frequency before and after the node

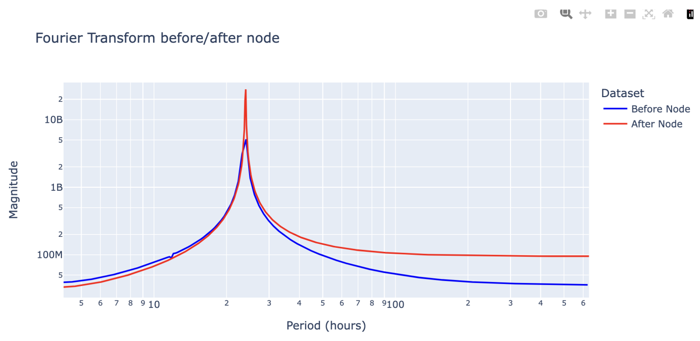


Figure 5: FFT Transform before and after node

3 Methodology

3.1 Problem formulation

Each timestamp records the *node* for two directions: North-South (NS) and East-West (EW). The objective is to identify these *nodes* with a 6 time-steps tolerance. We treat this task as **two separate multi-class classification tasks**, one for each direction, with the following classes:

Table 2: Node Classifications for EW and NS Directions

Direction	Classes
NS	Nothing, SS-NK, SS-CK, SS-HK, SS-EK, ID-NK, IK-CK, IK-HK, IK-EK
EW	Nothing, SS-NK, SS-CK, SS-HK, SS-EK, ID-NK, IK-CK, IK-HK, IK-EK, AD-NK

Here, 'Nothing' means "no node". While it might be possible to approach this as a single multi-class classification task, simultaneous events in both directions justify handling them as two separate problems. Indeed, the fact that a classifier can only give one prediction per timestamp would inevitably allow us to miss the second event. More details on this decision are detailed in the Model Separation and Recombination Strategy.

3.2 Model choice

Our main model is a **gradient-boosted trees** using **CatBoost** python package. In this section, we explain the rationale behind this decision. Since the size of the training dataset is small, we can immediately rule out some of the most data-hungry architectures such as transformer based models [3] or large neural networks. This leaves us the choice between gradient-boosted trees, and shallow neural networks. This decision requires more reflection.

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.

Table 3: Comparison of Gradient-Boosted Trees and Neural Networks in Multi-class Classification

Considering that gradient-boosted trees are faster to train, less prone to over-fitting, resources efficient, robust against outliers, and explainable, and that they had tremendous success in the competitive data science community, we chose them [4, 5].

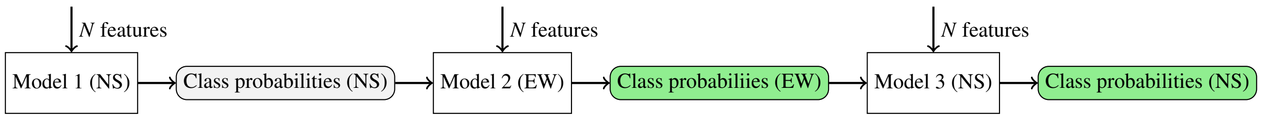
3.3 Model Separation and Recombination strategy

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:

	Single Model	Two Independent Models
Pros	• Accounts for NS-EW interaction • Simpler development and monitoring	• More control over precision and recall per direction • Faster training (can be parallelized) • Can predicts two events simultaneously
Cons	• Reduced control over individual direction metrics • Difficulty managing 16 classes simultaneously • Limited to one prediction per timestamp (if considered as multi-class problem)	• Assumes independence of EW and NS • Longer development due to individual tuning

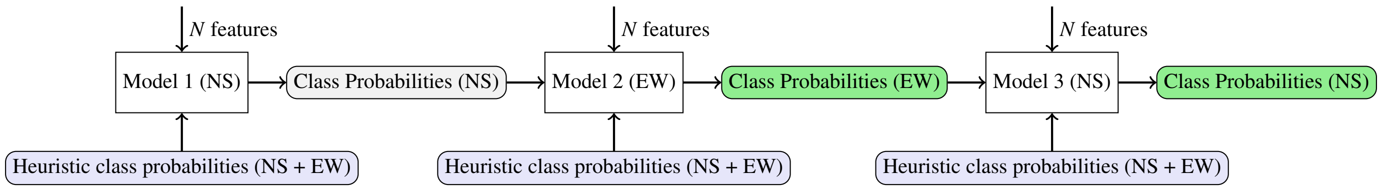
Table 4: Pros and cons of model separation strategies

Recombination Strategy: Our approach merges the benefits of independent predictions with insights into the interactions between NS and EW. This is achieved through a sequential prediction mechanism: **Model 1** initially predicts for NS using the N features created; **Model 2** then incorporates Model 1's outputs and the N features to predict for EW; finally, **Model 3** refines NS predictions using the outputs from Model 2 and the N features.



3.4 Model Stacking

Model stacking involves using multiple predictive models sequentially to enhance prediction accuracy. The heuristic model provided by challenge organizers is a method of characterizing the station-keeping and propulsion features of a satellite's PoL by tracking changes in the satellite's longitude and inclination waveform. This approach is somewhat orthogonal to tree-based models and can diversify our modeling. Therefore, we added the heuristic model to the previous pipeline, incorporating the predictions of the heuristic method as additional input to the tree-based model, as illustrated in the following diagram:



3.5 Model Prediction Strategy

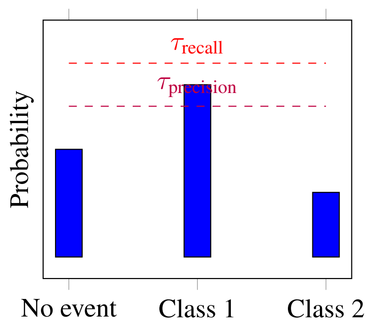
This section outlines our prediction strategy to obtain class from probabilities generated by the model, focusing on key evaluation metrics. A key observation is the F_2 score's emphasis on Recall, weighting it four times more than Precision. This insight led to the development of a **dual-threshold mechanism** to have more control over the precision and recall trade-offs in multi-class classification, crucial for managing multiple classification thresholds in unbalanced datasets.

- **Highest Probability and Class** $(p_{\textbf{max}}, c_{\textbf{max}})$: Represents the model's most confident prediction. If p_{max} exceeds the **precision threshold** ($\tau_{\textbf{precision}}$), the model makes a prediction, enhancing precision. If $p_{\text{max}} < \tau_{\text{precision}}$, the prediction is set to "no event" due to insufficient confidence.
- **Second Highest Probability and Class** $(p_{\textbf{second max}}, c_{\textbf{second max}})$: Represents the next most probable class. If c_{max} is "no event" and $p_{\text{second max}}$ exceeds the **recall threshold** ($\tau_{\textbf{recall}}$), 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.

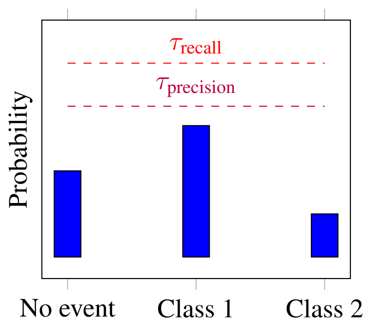
Parameter values are provided for reproducibility. It is noteworthy that the τ_{recall} is set higher for the NS direction compared to EW. This adjustment compensates for the higher number of classes in the EW (10) than in the NS (7) direction, requiring more stringent measures to enhance recall. Indeed, the more classes we have, the less the model is able to correctly identify each individual class.

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

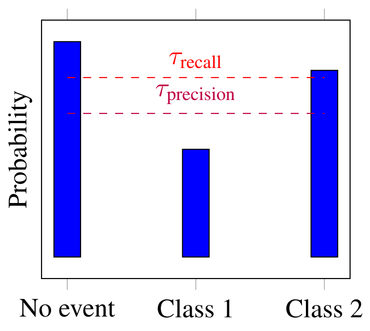
Table 5: Sensitivity Thresholds



(a)



(b)



(c)

Figure 6: Illustration of double-thresholding mechanism

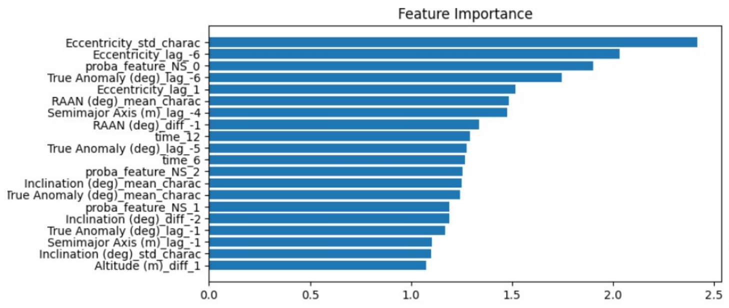
In figure (a), $Class\ 1$ is the most probable and is above the threshold $\tau_{\text{precision}}$. Thus, we predict $Class\ 1$. In figure (b), $Class\ 1$ is again the most probable, but it is below the $\tau_{\text{precision}}$ threshold. Therefore, this time we predict *No event*. In figure (c), *No event* is the most probable, however, $Class\ 2$ is above the τ_{recall} threshold. Consequently, we predict $Class\ 2$.

In summary, the **dual-threshold** mechanism allows us to effectively and easily control precision and recall in multi-class classification, significantly increasing the performance.

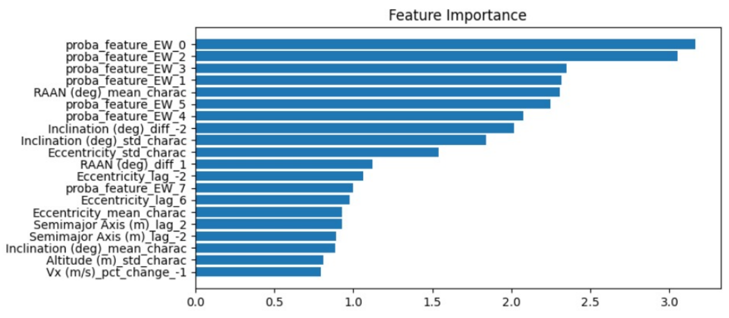
4 Core Observations and Reproducibility

4.1 Features Importance

Feature importance plots, illustrated below, provide valuable insights into the predictive power of specific features and validate the effectiveness of our research directions. We use the high interpretability of decision trees [6] to confirm the efficacy of our modeling choices.



(a) Model 2 (prediction of EW node)



(b) Model 3 (prediction of NS node)

Figure 7: Top 20 feature importance for each model

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" (see Section 2.2) highlights the benefits of pre-informing the model about the satellite types it is analyzing.

4.2 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 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.

Hardware	Type
CPU	Intel Xeon E5-2699 v4 (22 cores / 44 threads)
RAM	256 GB
Storage	512 GB SSD NVMe in PCI Express + 1 TB HDD

Model	Nb of features	Training Time
NS Model 1	321	1 hour
EW Model 2	321 + 9	3 hours
NS Model 3	321 + 10	1 hour

Training was conducted exclusively 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 training code and guidelines for replication are available here.

5 Conclusion

In this paper, we have introduced several innovative methods to improve AI performance in trajectory tracking. Our analysis demonstrated the significance of satellite-specific features and the predictive power of envelope variations. Through **frequency analysis**, we highlighted how changes in frequency are key predictors due to the seasonal and cyclical nature of satellite trajectories. Our approaches, including **Model Recombination** and **Model Stacking**, have effectively refined our predictions. Additionally, the implementation of a **dual-threshold mechanism** has optimized the precision-recall balance in multi-class classification.

Although we did not achieve the top position on the leaderboard, our model is notably well-suited for real-world applications. It combines simplicity, efficiency, fast training times, and robust generalization capabilities, distinguishing it as a prime candidate for practical deployment. With more time, we anticipate that feature engineering, particularly in frequency analysis, would further improve our model.

References

[1] 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.

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

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

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

[5] 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.

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