

LUNAR BROWSER UTILIZATION OF MACHINE LEARNING FOR TRAJECTORY SOLUTION PRODUCTION

Dylan N. Morrison-Fogel,^{*} Anthony L. Genova,[†] Paul G. Levinson-Muth,[‡]
and Regina Blue[§]

This paper describes the application of machine learning tools to produce Earth-Moon spacecraft trajectories with applications to NASA’s Commercial Lunar Payload Services (CLPS) and *Artemis* Human Landing System (HLS) programs. Existing trajectory solutions are used to train and test machine learning models to predict essential details of a trajectory sequence from Earth-launch to Low-Lunar Orbit, populating a database of solutions with future launch dates. The machine learning model will implement hyperparameter optimization for further re-training to improve model performance. Accurate predictive models decrease the time required to produce solutions and are readily implemented in the Lunar Browser tool.

INTRODUCTION

Space Policy Directive-1, issued in December 2017 states that, under the National Aeronautics and Space Administration (NASA) *Artemis* program, humans will return to the Moon for ground-breaking scientific discovery and as preparation for interplanetary crewed missions. The program is intended to be innovative and sustainable with commercial and international partners, beginning with the return of humans to the Moon for long-term exploration.¹ NASA’s Commercial Lunar Payload Services (CLPS) initiative aims to partner with private companies to develop and operate lunar landers, particularly in the polar regions as part of the *Artemis* program.² CLPS missions deliver science and in situ resource utilization technology to the Moon’s surface to provide NASA with new insights about the lunar environment on both the near and far side.³ Of particular interest are several sites in the south pole region, specifically Malapert A, Shackleton Connecting Ridge, and Mons Mouton, as well as various mid-latitude sites such as Reiner Gamma, Mare Crisium, and Schrödinger Basin.^{4,5,6,7,8,9,10}

The fields of Artificial Intelligence (AI) and Machine Learning (ML) have experienced an increase in utility and feasibility for many data science applications. This has prompted the President’s 2025 Budget Request to allocate funds for improving test efficiency and advancing space exploration using AI and ML tools.¹¹

Lunar Browser (LB) is a tool currently in development which aims to provide trajectory solutions for an Earth-Moon transfer.^{12,13} A database of existing trajectory solutions can be accessed to provide initial guesses for trajectory solutions not yet included in the database, with all new

^{*} Spaceflight Dynamicist, NASA Ames Spaceflight Division, Axient, 94035.

[†] Astrodynamics Subject Matter Expert, NASA Ames Spaceflight Division, NASA, 94035

[‡] Flight Systems Engineer, NASA Ames Spaceflight Division, NASA, 94035.

[§] CLPS Deputy Programmatic Manager, NASA Johnson Space Center, NASA, 77058

solutions being automatically appended to the main database. The existing LB database consists of over 220,000 Earth-Moon trajectory solutions with launch dates between January 1st, 2025, and January 1st, 2030. The rate at which further solutions are generated may improve with the application of ML techniques.

Extreme Gradient Boosting (XGBoost) is a machine learning library which can be applied to the LB database for regression, classification, ranking, and more. XGBoost has demonstrated superior performance in data science competitions¹⁴ and with its comparatively fast training times, is well suited for LB's ever-growing solution database. Beyond utilizing ML algorithms for populating solutions, XGBoost is also capable of identifying trends which are not immediately apparent when directly applying regression methods to the data, revealing hidden subsets of trajectories.

PROBLEM STATEMENT

LB's database continues to grow in terms of the types of trajectory solutions and the launch date range for which these solutions encompass. Leveraging the established trajectory database, implementation of ML techniques will allow for improved efficiency in populating further solutions. Rapid solution generation may take two forms, 1) ML models trained on existing data will provide solutions for future launch dates, or 2) the ML models provide comparable "initial guesses" to the solvers for which detailed trajectory information can be extracted. The current LB solution generation process requires a manual search of reference trajectories from the existing LB database, represented by a hexadecimal ID. Specifying desired parameters of the trajectory and utilizing a trained model to provide parameters used to construct a successful trajectory will replace the time-consuming method of manual user searches.

Training a ML model on the existing dataset will require specific data preprocessing, an analysis of the input features and optimization of the model hyperparameters. Success of ML implementation can be measured by comparing test results to a validation subset. Additionally, for the purposes of LB, success can be measured by comparable or reduced time required to generate trajectory solutions.

SOFTWARE TOOLS, ASSUMPTIONS & CONSTRAINTS

The primary tool used for calculation and propagation of trajectories was ANSYS Systems Tool Kit (STK). Propagators utilize an 8th/9th order Runge-Kutta numerical integrator for orbit propagation. Force models included a N-body gravity field (50X50 spherical harmonics Earth and Moon gravity fields and point mass assumption for the remaining planets). The propagator also assumed thermal and solar radiation pressures, and a Jacchia-Roberts atmospheric model at Earth.

Driving STK is the Lunar Browser tool, a collection of Python scripts and data files which serve as a user interface as well as an interface between the trajectory database and STK. LB extracts trajectory segment data from STK, allowing users to plot necessary trajectory data as well as recreate the entire trajectory in the STK user interface; an example of a direct Earth-Moon transfer is shown in Figure 1. Further, LB acts as a trajectory solution generator for single or large numbers of trajectories, allowing users to generate specific trajectory types over a large span of launch dates. LB can adapt to a variety of trajectory types although it is primarily populated by solutions to direct Earth-Moon transfers from Low-Earth Orbit (LEO) to Low-Lunar Orbit (LLO), and for the purposes of this paper, will use direct transfers to train the ML model. The characteristic constraints for this trajectory type are outlined in Table 1.

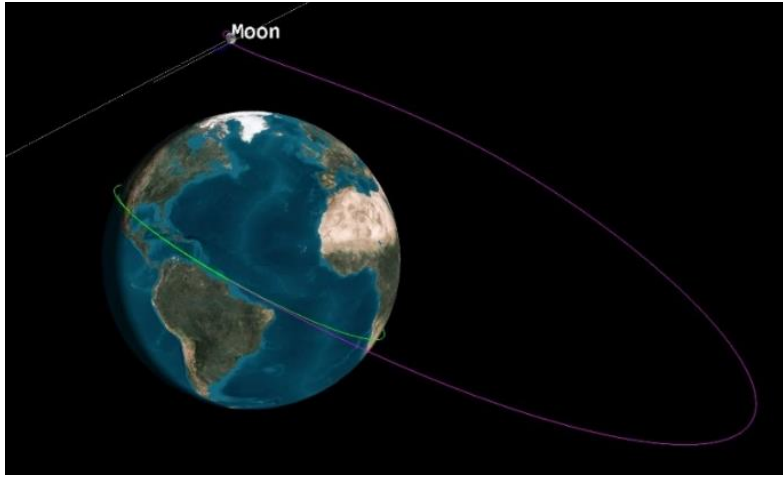


Figure 1. Lunar Browser sample solution for an Earth-Moon trajectory.

Table 1. Description and constraints of the model input features.

Trajectory Feature	Description	Constraint
Launch Date	UTCG date of launch from Earth	January 1 st , 2025, to January 1 st , 2030
Launch Site	Launch site to establish geodetic initial conditions	NASA Kennedy Space Center
Coast Inclination	Inclination (Earth ICRF) of the coast segment following ascent burnout	28.62° or 35°
Coast Duration	Duration of time between launch vehicle first-stage burnout and Trans-Lunar Injection (TLI)	$0 \leq t \leq 120$ (min)
Pre-LOI B-Theta	Determines the heading in which Lunar-Orbit Insertion (LOI) is performed (Lunar North/South/East/West)	-90°, 90°, 0°, 180°
Duration TLI-LOI	Duration between TLI burn and LOI burn	4 – 14 days (1-day increments)
LLO Parameters	Final low-lunar orbit achieved by the spacecraft	100 km altitude, circular ($e = 0$) lunar orbit

DATABASE EXPANSION

LB currently generates new trajectory solutions through direct interaction with the user, requesting a set of constraints and the hexadecimal ID of a previous solution. LB will retrieve line items from its existing database corresponding to the provided hexadecimal ID and pass them to the appropriate parameter entries in the Mission Control Sequence (MCS) of a designated spacecraft object in STK. STK will then run a MCS of the specified spacecraft object, converging on a trajectory solution for which segment data is extracted and appended to the LB database under a new hexadecimal ID. Convergence must occur within the user-defined number of iterations. While this process is adequate to generate tens of solutions with user supervision, initial guesses not sufficiently close to the desired solution condition may take numerous iterations, increasing the time required for expanding the database when hundreds or thousands of solutions are desired. Further, solutions

within the existing LB database must be sufficiently close in epoch and trajectory segment parameters for the MCS to converge. As a result, growing the solution database into unexplored solution spaces requires iteratively building on the existing set until reaching a sufficiently close set of solutions to the desired trajectory space.

Current Solution Space

Trajectory segments within the MCS consist of Launch, LEO Coast, TLI, Lunar approach, LOI, and LLO, illustrated in Figure 2. Refinements for mission architectures and capabilities (e.g., phasing loops from Earth or dividing LOI into multiple burns based on propulsion system capabilities) fall beyond the scope of this paper.

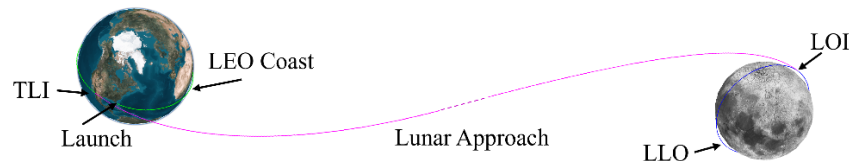


Figure 2. An Earth-Moon trajectory consisting of a Launch from KSC, Coasting in LEO, a TLI maneuver, coasting segment to lunar approach, a LOI maneuver, finally achieving the desired LLO.

Input features for the XGBoost model are those which can be readily provided as inputs from a LB user, specifically Launch Date, LEO Inclination, B-Theta angle on lunar approach, and the Time-of-Flight (TOF) between TLI and LOI. Figure 3 illustrates the orientation of the B-Theta angle with respect to the incoming trajectory, allowing the targeter to converge on Northern, Eastern, Western, or Southern approaches. The Lunar Browser database includes all approach directions, however for the purposes of this paper, only Northern and Southern approaches will be considered assuming a B-Theta angle of -90 or 90, respectively.

A list of targets for the XGBoost model to return are those which can be combined with the input features to construct a fully defined STK MCS profile. These targets are Launch Azimuth, Launch Burnout Velocity, LEO Coast Duration, TLI C_3 , TLI Declination of Launch Azimuth (DLA), TLI Velocity Azimuth, TLI Delta-V, and LOI Delta-V. The STK MCS profile also relies on inputs such as Launch Latitude, Longitude, and Altitude, which currently do not vary between existing solutions in the LB database. As the XGBoost model uses the input feature data to explain variations in the target features, it is not necessary to model these currently static MCS parameters.

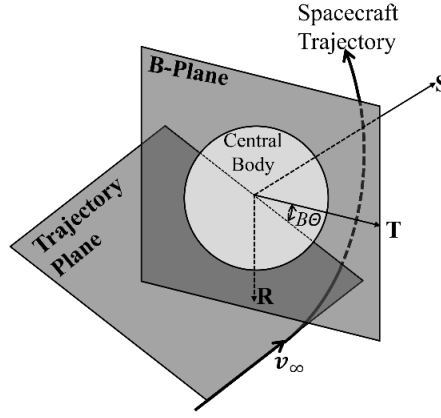


Figure 3. Illustration of the B-Plane coordinate system.

METHODS

Specifying the correct set of hyperparameters when initializing a XGBoost model is essential to the model training and performance, each set being unique to the output desired. The *hyperopt* python library implements a Bayesian optimization strategy to iteratively assess the best hyperparameter set over a given hyperparameter search space.¹⁵ The hyperparameter search space is outlined in Table 2.

Table 2. Hyperparameter description and evaluated ranges for the XGBoost regressor.

Hyperparameter	Description	Range
'max_depth'	Maximum depth of the decision tree	3 - 10 (uniformly distributed)
'gamma'	Minimum loss reduction required for partition on a leaf node of decision tree	1 - 9 (uniformly distributed)
'reg_alpha'	L1 regularization term on weights (increasing makes the model more conservative)	0 - 10 (uniformly distributed)
'reg_lambda'	L2 regularization term on weights (increasing makes the model more conservative)	0 - 10 (uniformly distributed)
'colsample_bytree'	Subsample ratio of columns when constructing each tree	0.5 - 1 (uniformly distributed)
'min_child_weight'	Minimum sum of instance weight needed in a child	0 - 10 (uniformly distributed)
'learning_rate'	Step size shrinkage used in update to prevent overfitting	0.001 - 1 (uniformly distributed)
'n_estimators'	Number of possible trees in the decision tree model	300 - 2000 (randomly selected integers)
'min_samples_split'	Minimum number of samples required to split an internal node	1 - 20 (uniformly distributed)
'min_samples_leaf'	Minimum number of samples required to be at a leaf node	1 - 10 (uniformly distributed)

Using the existing LB database, the *hyperopt* analysis is performed for each model and the intermediate and optimal results recorded. Another simple XGBoost classification model is used as a “wrapper” and trained on each of the intermediate results. Shapley additive explanations (SHAP) values leverage game theory to assign relative values of importance to model features.¹⁶ For the purposes of this analysis, SHAP values of the models show which hyperparameters are of greatest importance and influence on the ability to properly train each XGBoost model. Hyperparameters found to have minimal feature importance do not need to be included in model retraining, reducing model complexity and training time.

The sets of optimal hyperparameters are then used to train the final XGBoost model for each desired output parameter. Training and testing are performed using the existing LB database and an objective function seeks to minimize the overall root mean squared error (rmse) between the predicted set and the test set. Provided a set of input features extracted from sample trajectories in the LB database, trained models predict MCS parameters which are directly introduced to a STK scenario object and the MCS is run. Solution generation with the described method is compared against the LB trajectory solution generator over an identical trajectory solution sample set.

RESULTS

Hyperparameter sets across the XGBoost models implementing *hyperopt* show *learning_rate*, *gamma*, *reg_alpha*, *n_estimators*, and *max_depth* have greatest feature importance when training. Figure 4 - Figure 10 are provided in order of greatest SHAP values (ranked: LOI Delta-V, TLI DLA, TLI Velocity Azimuth, Coast Duration, TLI Delta-V, Launch Azimuth, TLI C₃). Each plot illustrates the importance of each hyperparameter to the XGBoost model and therefore the sensitivity of the model to variations in hyperparameter set.

Figure 4 shows the LOI Delta-V model is impacted largely by *learning_rate* with a positive SHAP value, indicating predictions are shifted to a greater positive outcome with variations in this hyperparameter. Further, *reg_alpha* and *n_estimators* hold a high feature importance for the model but have negative SHAP values, indicating a sensitivity to these hyperparameters which shift the model away from an improved model (greater rmse).

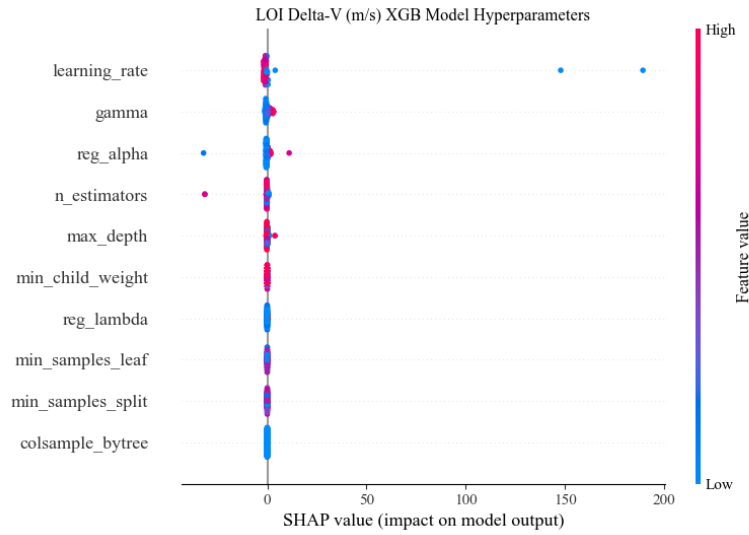


Figure 4. LOI Delta-V Model Hyperparameter SHAP Values Indicating 'learning_rate' has Greatest Model-Training Feature Importance.

Figure 5 shows the TLI DLA model is also impacted largely by *learning_rate* with a positive SHAP value, again indicating predictions are shifted to a greater positive outcome with variations in this hyperparameter. The *n_estimators* hyperparameter holds a high feature importance for the model but has negative SHAP values, indicating it may negatively impact model fit.

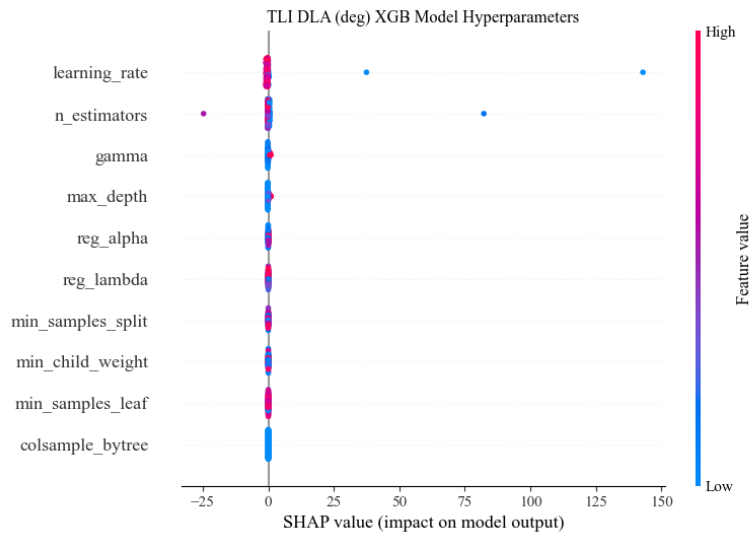


Figure 5. TLI DLA Model Hyperparameter SHAP Values Indicating 'learning_rate' has Greatest Model-Training Feature Importance.

Figure 6 shows the TLI Velocity Azimuth model is also impacted largely by *gamma*, *learning_rate*, and *n_estimators*, with positive SHAP values, indicating predictions are shifted to a greater positive outcome with variations in this hyperparameter. It should be noted that *gamma* values which are of higher feature importance have a positive impact on the model training while values of lower feature importance have an adverse impact on the model fit. This suggests the *gamma* parameter is particularly beneficial to the model fitment. Again, the *n_estimators* hyperparameter holds a high feature importance for the model but has negative SHAP values, indicating it may negatively impact model fit.

Figure 7 illustrates the Coast Duration model is also impacted largely by *gamma*, *learning_rate*, and *max_depth*. Increased feature importance in *learning_rate* corresponds to a negative impact on the model fit, while increased feature importance in *gamma* and *max_depth* correspond to positive impacts. It should be noted that the Coast Duration model is not sensitive to *n_estimators* and its sensitivity is symmetrical over the spectrum of feature values, indicating high feature importance benefits the model while low feature importance does not benefit the model. This symmetry also suggests the *n_estimators* parameter is particularly beneficial to the model fitment.

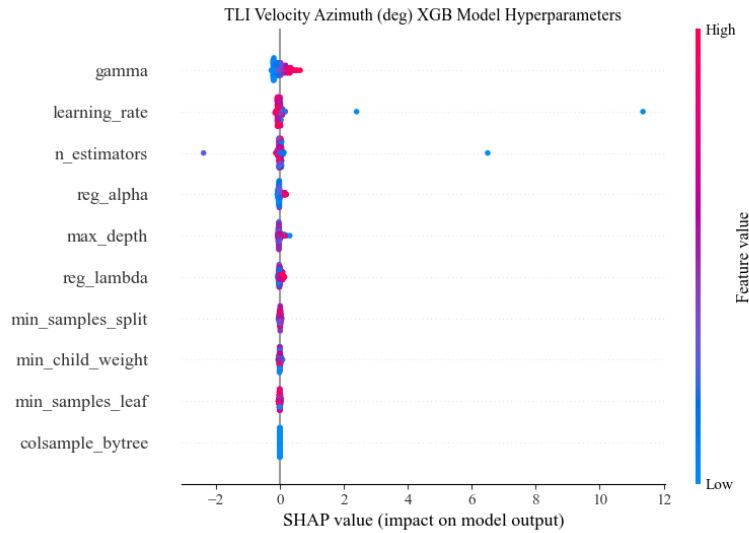


Figure 6. TLI Velocity Model Hyperparameter SHAP Values Indicating 'gamma' has Greatest Model-Training Feature Importance.

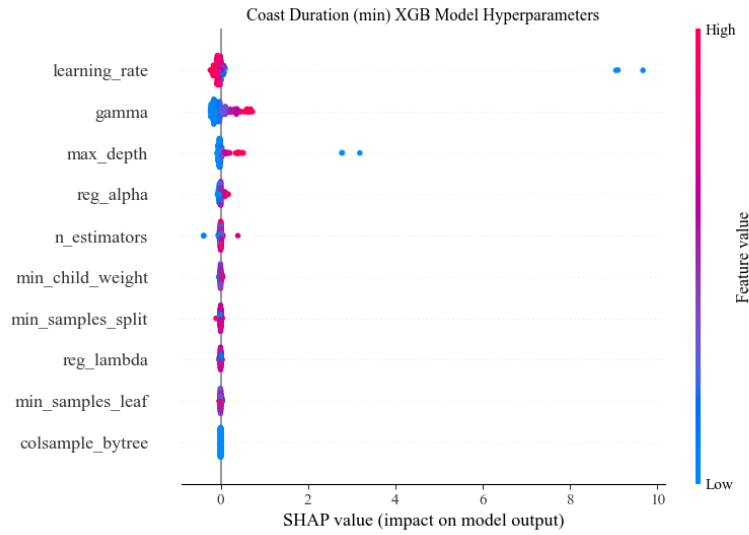


Figure 7. Coast Duration Model Hyperparameter SHAP Values Indicating 'learning_rate' has Greatest Model-Training Feature Importance.

Figure 8 demonstrates the TLI Delta-V model is also impacted largely by *gamma*, *learning_rate*, *n_estimators*, and *max_depth*, with additional feature importance placed on *reg_alpha*. Increased feature importance in *gamma* corresponds to a positive impact on the model fit, while decreased feature importance in *gamma* corresponds to negative impacts.

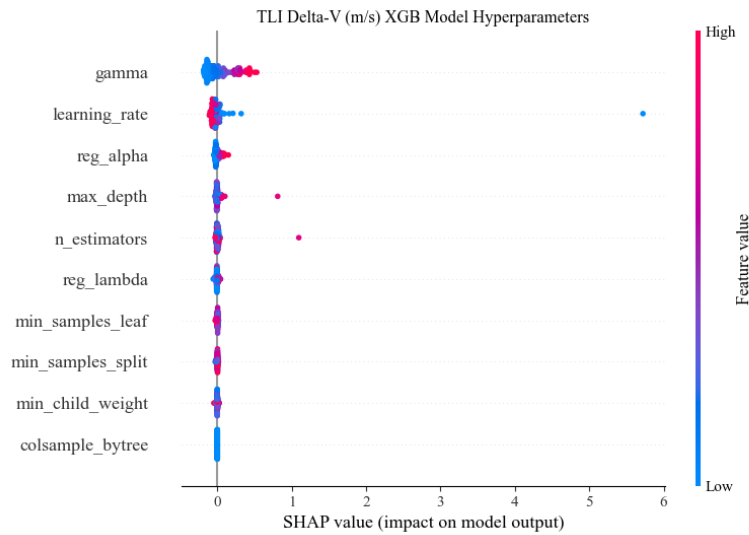


Figure 8. TLI Delta-V Model Hyperparameter SHAP Values Indicating 'gamma' has Greatest Model-Training Feature Importance.

As Figure 9 reveals, the Launch Azimuth model is impacted largely by *gamma*, *reg_alpha*, *learning_rate*, and *max_depth*, with additional feature importance placed on *reg_lambda* and *n_estimators*. The plot displays a gradient dispersal of the feature significance across the SHAP values, implying strong correlation. However, it should be noted that the relative SHAP values in this plot are magnitudes lesser than the other model SHAP values.

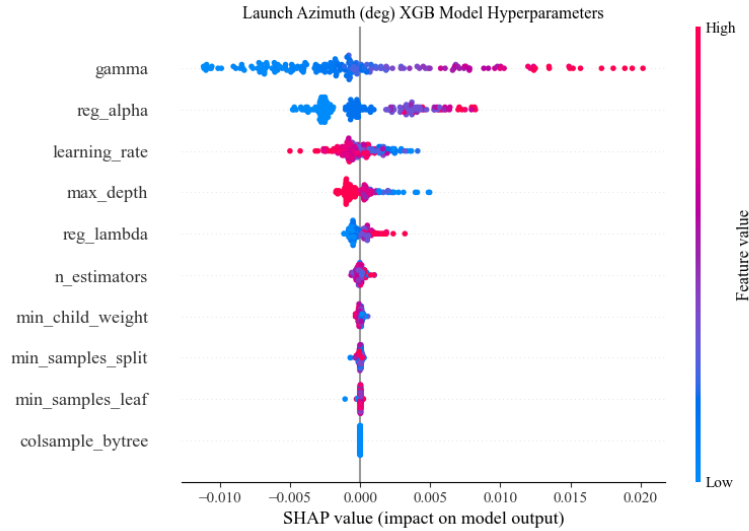


Figure 9. Launch Azimuth Model Hyperparameter SHAP Values Indicating 'gamma' has Greatest Model-Training Feature Importance.

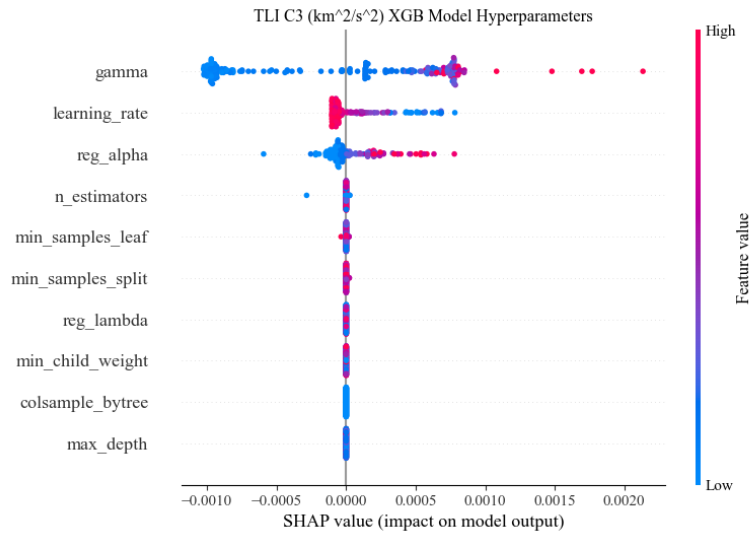


Figure 10. TLI C₃ Model Hyperparameter SHAP Values Indicating 'gamma' has Greatest Model-Training Feature Importance.

Figure 10 shows the TLI C₃ model is impacted largely by *gamma*, *reg_alpha*, and *learning_rate*. The plot again displays a gradient dispersal of the feature significance across the SHAP values,

implying correlation but again, it should be noted that the relative SHAP values in this plot are magnitudes lesser than the previous models.

Table 3 exhibits the optimal hyperparameters for each model, providing a guide for future XGBoost model training on the LB database solution set. The value for *n_estimators* can range between 300 and 1250 but should not be below the lesser to avoid undertraining and exceed the greater to reduce training time and to avoid overtraining. Through all models, the *learning_rate* remains relatively high, indicating subsequent models incorporate a majority of each new decision tree in the ensemble. The *max_depth* hyperparameter covers the full range of its search space, representing small and large decision trees depending on the model being trained. The minimum loss reduction, *gamma*, remains low across all models, suggesting there is minimal need to prevent model overfitting, with lower values being less conservative. Penalization of the features which increase the objective function (rmse) with *reg_alpha* varies throughout the models, with low values in all but the TLI DLA model, which implements a relatively high penalization. The *reg_lambda* values in the Coast Duration, TLI DLA, and TLI Velocity Azimuth models are large, suggesting the models need to be more conservative by encouraging smaller weights on the decision tree leaves. The *min_child_weight* values in TLI DLA and the LOI Delta-V models are large, allowing less complexity to the decision tree branches and leading to a more conservative model. Conversely, the *min_child_weight* values in TLI Velocity Azimuth and TLI Delta-V are low, permitting increased tree complexity. In all models, *col_sample_bytree* remains zero and the model limits the number of features used to build decision trees, with sufficient insight from the primary data features. The minimum samples required to split an internal node, controlled by *min_samples_split*, is relatively high throughout the models and creates a more conservative model, though it is not so high that overfitting occurs. The final hyperparameter, *min_samples_leaf*, is large in all models except the TLI C₃ model, suggesting most of the models tend to conservatism.

Table 3 - Optimal Hyperparameter Set for Each Trajectory Parameter Model

Hyperparameter	Launch Az.	Coast Duration	TLI C ₃	TLI DLA	TLI Vel. Az.	TLI Delta-V	LOI Delta-V
' <i>n_estimators</i> '	1066	807	1097	932	1250	723	300
' <i>learning_rate</i> '	0.836	0.999	0.984	0.823	0.906	0.997	0.918
' <i>max_depth</i> '	10	4	4	3	6	5	10
' <i>gamma</i> '	1.019	1.291	1.008	1.215	2.466	1.005	1.004
' <i>reg_alpha</i> '	0	2	0	9	0	0	0
' <i>reg_lambda</i> '	1	6	0	9	4	1	2
' <i>min_child_weight</i> '	3	3	4	10	0	1	9
' <i>col_sample_bytree</i> '	0	0	0	0	0	0	0
' <i>min_samples_split</i> '	10	16	11	10	10	19	13
' <i>min_samples_leaf</i> '	7	7	3	10	9	9	6

The number of successful solutions generated by the XGBoost model are comparable with the LB trajectory solution generator. Of 200 sample solutions, 36 iterations of the LB produced MCS parameters did not converge on a solution. With MCS input parameters produced by the XGBoost model, 37 solutions did not converge. Figure 11 plots the resulting convergence time against iteration number for both the XGBoost model and LB solution generator. Most of the solutions for both XGBoost and LB converge in two to three seconds. The plot illustrates convergence times on

XGBoost solutions are similarly distributed to the convergence times for the LB solution generator. After the 200 iterations were complete, the cumulative time for XGBoost solutions was 421 seconds, with 409 seconds for LB.

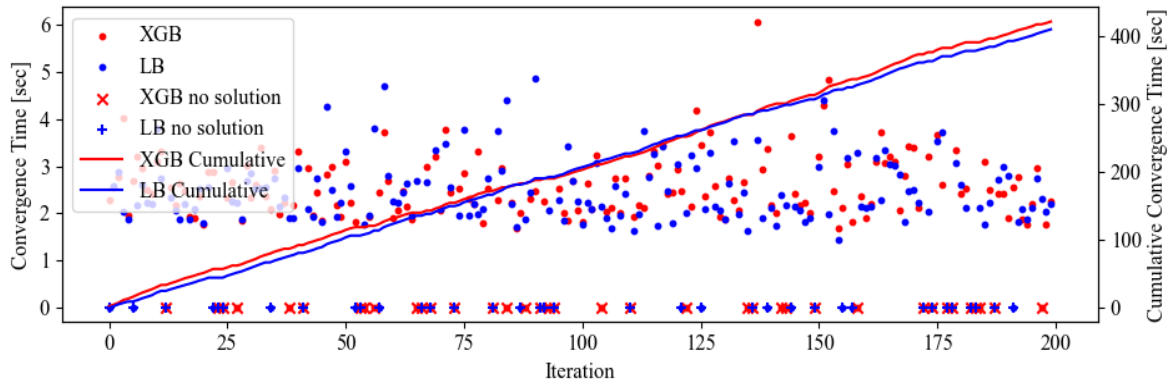


Figure 11. STK Convergence Time Provided MCS Parameters from XGBoost Models and LB Database.

Modeled XGBoost results, plotted against the LB trajectory generator results in Figure 12, do not show significant outliers and display similar trends across the iterations. XGBoost has shown proficiency in trend representation, particularly with respect to Coast Duration, TLI Delta-V, and LOI Delta-V.

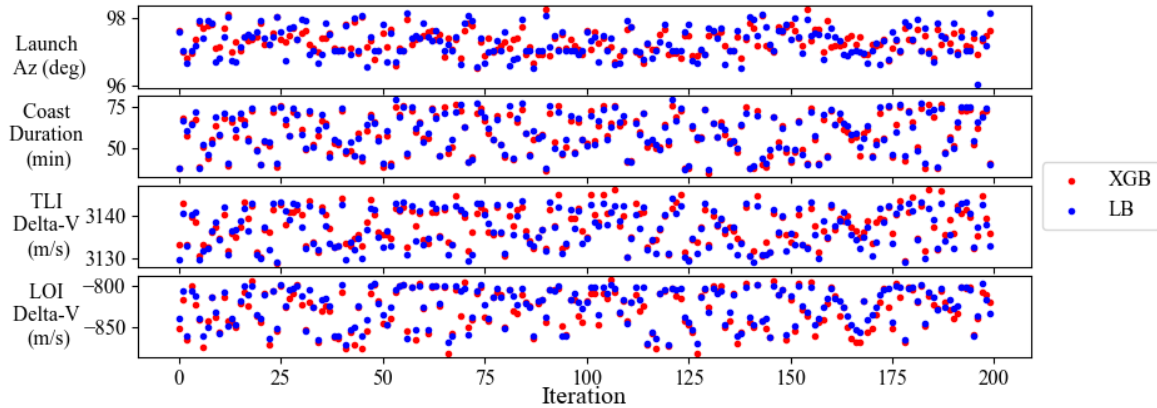


Figure 12. STK MCS Input Parameters from XGBoost Models and LB.

Figure 13 provides the SHAP plot for input features used to train the model for LOI Delta-V. The Julian Date (launch epoch) demonstrates the greatest feature importance, with higher feature values being centered around a zero SHAP value. The Pre-LOI B-Theta feature for this model has a distribution of high and low feature value across positive and negative regions as a result of targeting either Northern (-90°) or Southern (90°) poles on lunar approach. SHAP values for the Duration between TLI and LOI show trajectories with lower-duration values are increasingly difficult for the model to represent. Coast Inclination has the lowest feature importance but shows lower feature values (28.6°) are easier to represent with the XGBoost model when compared to higher feature values (35°).

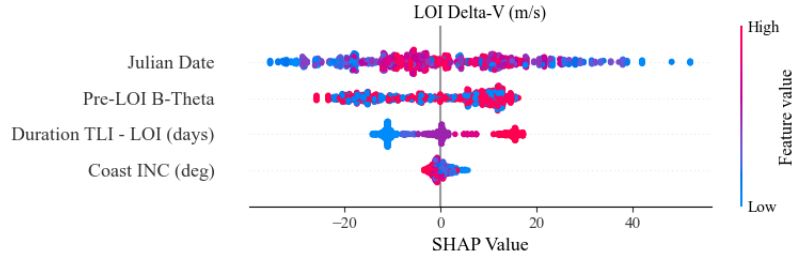


Figure 13. XGBoost Model SHAP Values Indicating Julian Date (Launch Epoch) has the Greatest Feature Importance in Predicting LOI Delta-V.

Figure 14 provides the SHAP plot for input features used to train the model for TLI DLA. The Pre-LOI B-Theta feature for this model holds greatest feature importance and again has a distribution of high and low feature value across positive and negative regions as a result of targeting lunar poles. Consistent with Figure 13, SHAP values are lower for reduced TLI to LOI Durations and difficult for the model to represent. Similarly, Coast Inclination shows lower feature values are easier to represent when compared to higher feature values.

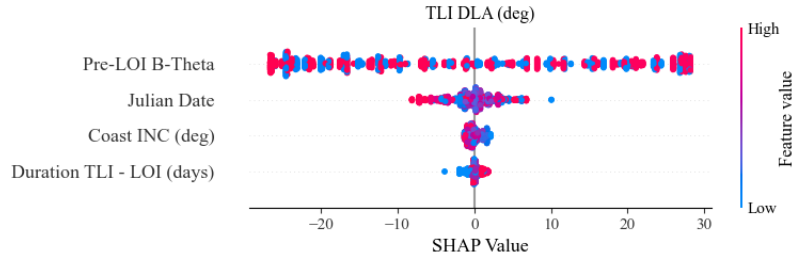


Figure 14. XGBoost Model SHAP Values Indicating Lunar Approach B-Theta has the Greatest Feature Importance in Predicting TLI DLA.

Figure 15 provides the SHAP plot for input features used to train the model for Coast Duration. The Pre-LOI B-Theta feature for this model again holds greatest feature importance and similarly has a distribution of high and low feature value across positive and negative regions as a result of targeting lunar poles. Unlike Figure 13, SHAP values are higher for reduced TLI to LOI Durations and easier to model.

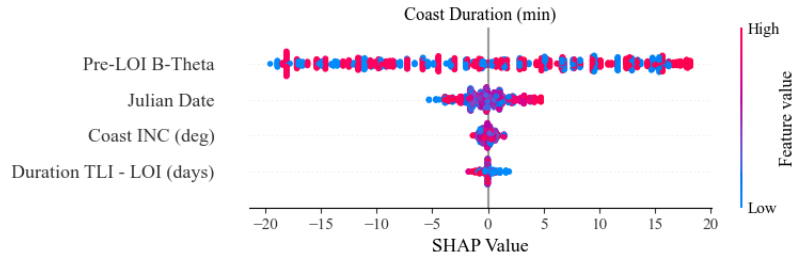


Figure 15. XGBoost Model SHAP Values Indicating Lunar Approach B-Theta has the Greatest Feature Importance in Predicting Coast Duration.

Figure 16 the SHAP plot for input features used to train the model for Coast Duration and similarly, the Pre-LOI B-Theta feature for this model holds greatest feature importance and has a distribution of high and low feature value across Northern and Southern lunar poles. Lower values of Coast Inclination have negative SHAP values while higher feature values remain positive.

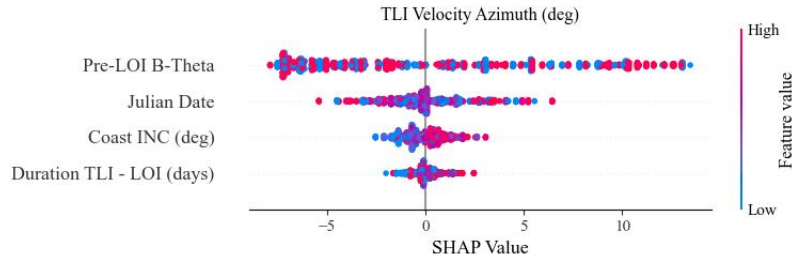


Figure 16. XGBoost Model SHAP Values Indicating Lunar Approach B-Theta has the Greatest Feature Importance in Predicting TLI Velocity Azimuth.

Figure 17 provides the SHAP plot for input features used to train the model for TLI Delta-V and similarly, the Pre-LOI B-Theta feature for this model holds greatest feature importance and similarly has a distribution of high and low feature values across SHAP values. Lower values of Coast Inclination have negative SHAP values while higher feature values remain positive.

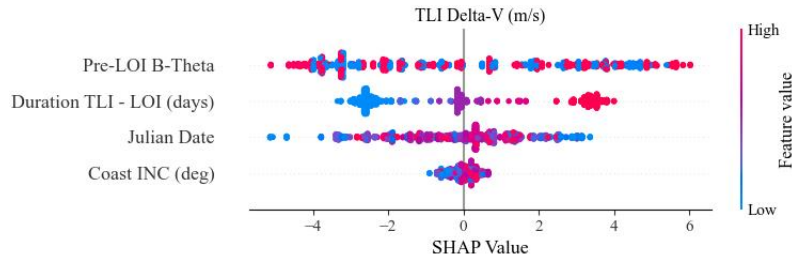


Figure 17. XGBoost Model SHAP Values Indicating Lunar Approach B-Theta has the Greatest Feature Importance in Predicting TLI Delta-V.

Figure 18 provides the SHAP plot for input features used to train the model for Launch Azimuth with Coast Duration holding greatest feature importance. Lower values of Coast Duration have negative SHAP values while higher feature values remain positive.

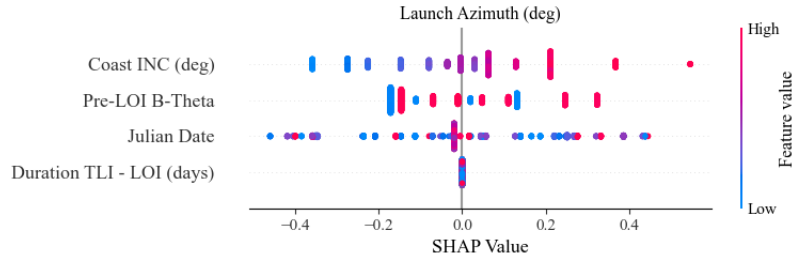


Figure 18. XGBoost Model SHAP Values Indicating LEO Coast Inclination has the Greatest Feature Importance in Predicting Launch Azimuth.

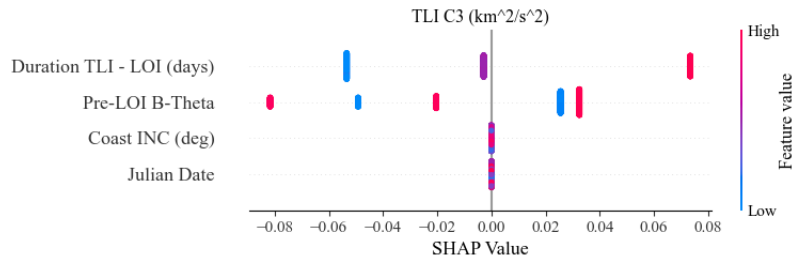


Figure 19. XGBoost Model SHAP Values Indicating the Duration Between TLI and LOI has the Greatest Feature Importance in Predicting TLI C₃.

Figure 19 provides the SHAP plot for input features used to train the model for TLI C₃ showing the Duration between TLI and LOI holds greatest feature importance. The Duration between TLI and LOI in Figure 18, as well as Coast Inclination and Julian Date (launch epoch) in Figure 19, show a consistent SHAP value of zero, indicating there is no impact on the model prediction as these values change. Figure 20 further illustrates the effect of training a model to fit invariant data. The LB database consists exclusively of trajectories launching from KSC (specified in Table 1) and, as expected, no variation in Launch Altitude occurs (all begin at the launch pad).

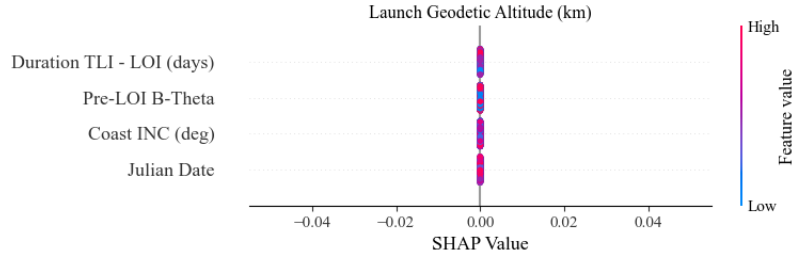


Figure 20. XGBoost Model SHAP Values Illustrating Zero Feature Importance Due to a Consistent Launch Site Across Database Solutions.

DISCUSSION

Hyperparameters examined after the *hyperopt* process presented optimal values indicate most models tend to be conservative, favoring reduced complexity of the overall model and abating overfitting. Despite their overall conservative similarities, the hyperparameter sets vary between each model, hence individual models should be used to predict MCS values rather than a multiple-output regression model. Hyperparameters like *col_sample_bytree*, with consistently low SHAP values between the models, are indicative of minimal importance on the model and can be ignored or unspecified when the model is retrained. As the database expands to a variety of other trajectory types and conditions, MCS parameters contained in the LB database will increase to include them. The XGBoost model is not capable of predicting anything beyond trends detected in the training data and retraining is required each time the number of distinct trajectory types increases.

The XGB model produced a comparable number of successful trajectory solutions to the LB trajectory solution generator. Although STK is still used to calculate the remaining trajectory parameters, MCS segments can be modeled through machine learning techniques, negating the requirement for time-intensive manual searches for appropriate hexadecimal identifiers in the LB database. Given the relative time for generating each solution is comparable to the process performed by LB, it is a viable method for populating the LB database with a high volume of results. Input features with greatest importance to trained models are Pre-LOI B-Theta, Launch Epoch, and Coast Inclination.

CONCLUSION

A method of Earth-Moon trajectory solution generation is presented which implements machine learning techniques to expedite database population. Leveraging the existing database of trajectories, a XGBoost regression model is trained to return key parameters which readily establish trajectories in a STK MCS profile. The XGBoost hyperparameters tuned using *hyperopt* are used to train a XGBoost model, effectively improving its overall accuracy. SHAP Value plots of the hyperparameter search space indicate the model is most influenced by the *learning_rate*, *gamma*, and *n_estimators*. The remaining hyperparameters exhibit minimal impact magnitude on the model output.

When compared to the Lunar Browser trajectory generation method, MCS parameters predicted by the tuned XGBoost model have similar convergence times and are equivalent in their number of successful MCS runs. With no requirement to manually search for and specify individual hexadecimal identifiers of existing LB trajectory solutions, employing XGBoost models to populate a set of MCS parameters for solution generation is a viable option to reduce the overall time to populate the Lunar Browser database.

The described machine learning method will be regularly updated alongside the continuously growing LB database, expanding to include a variety of launch sites, LEOs, and LLOs. Existing models are readily available for the generation of direct transfer solutions with launch dates beyond the 2027-2030 search space. Utilizing similar methods as those described in this paper, future machine learning models may be trained on diverse trajectory types (phasing loops, low energy, low thrust, etc.), enhancing the capabilities of the Lunar Browser tool.

ACKNOWLEDGEMENTS

The authors are appreciative of the support received across multiple NASA centers and programs including NASA Ames Research Center Spaceflight Division (notably the Flight Dynamics Team and the Mission Design Center), NASA CLPS program, and the NASA *Artemis* program.

REFERENCES

- ¹ P. Memorada, Memorandum on Space Policy Directive, Washinton DC: The White House, 2021.
- ² J. Rubio, "Commercial Lunar Payload Services Overview," NASA, 24 October 2023. [Online]. Available: <https://www.nasa.gov/reference/commercial-lunar-payload-services/>. [Accessed 27 August 2024].
- ³ J. Schonfeld, "Summary of the Contracted Deliveries of NASA Payloads to the Moon Via Commercial Lunar Payload Services (CLPS)". In *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* pp. 863-866. 2023.
- ⁴ "Commercial Lunar Payload Services Landing Sites." CLPS Landing Sites Graphic, NASA, 14 Feb. 2024, www.nasa.gov/image-detail/landing-site-graphic-updated-1-29-24/.
- ⁵ H. F. Haviland, R. C. Weber, C. R. Neal, P. Lognonné, R. F. Garcia, N. Schmerr, S. Nagihar, R. Grimm, D. G. Currie, S. Dell'Agnello, T. R. Watters. "The lunar geophysical network landing sites science rationale". *The Planetary Science Journal*. 3(2):40. 2022.
- ⁶ J. J. Parker, F. Dovis, B. Anderson, L. Ansalone, B. Ashman, F. H. Bauer, G. D'amore, C. Facchinetti, S. Fantinato, G. Impresario and S. A. McKim, "The lunar gnss receiver experiment (LUGRE)". In *Proceedings of the 2022 International Technical Meeting of The Institute of Navigation* (pp. 420-437). 2022.
- ⁷ D. T. Blewett, H. M. Meyer, M. J. Kinczyk, J. Halekas, G. C. Ho, B. T. Greenhagen, B. W. Denevi, R. L. Klima, J. T. Cahill, B. J. Anderson, and S. K. Vines. "Selection and Characterization of the Landing Site for NASA CLPS PRISM1 Lunar Lander Mission CP-11". In *54th Lunar and Planetary Science Conference (LPSC)*. 2023.
- ⁸ A. Aboobaker, M. Panning, D. Bugby. "The Farside Seismic Suite: A Novel Approach for Long-term Lunar Seismology." In *2024 IEEE Aerospace Conference* 2024.
- ⁹ D. Andrews, "VIPER: Systems integration status". In *74th International Astronautical Congress (IAC)*. 2023.
- ¹⁰ B. Baker-McEvelly, S. Bhadauria, D. Canales, and C. Frueh. "A comprehensive review on Cislunar expansion and space domain awareness". *Progress in Aerospace Sciences*. 2024.
- ¹¹ "FY 2025 President's Budget Request Summary." FY 2025 Budget Estimates, National Aeronautics and Space Administration, 13 Mar. 2024, www.nasa.gov/wp-content/uploads/2024/03/nasa-fy-2025-congressional-justification.pdf.
- ¹² A. L. Genova, D. N. Morrison-Fogel, P. G. Levinson-Muth, A. Dono Perez and R. Blue, "'Lunar Browser Trajectory Tool," in *Software for the NASA Science Mission Directorate Workshop*, Washington, DC, 2024.
- ¹³ A. L. Genova, D. N. Morrison-Fogel and P. G. Levinson-Muth, "Transfers from TLI to Lunar Frozen Orbits with Applications to NASA's CLPS & Artemis Programs," in *AAS/AIAA Space Flight Mechanics Meeting*, Orlando, 2024.
- ¹⁴ S. R, S. Sanjay Ayachit, V. Patil and A. Singh, "Competitive Analysis of the Top Gradient Boosting Machine Learning Algorithms," in *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 2020.

¹⁵ S. Putatunda and K. Rama. "A Modified Bayesian Optimization Based Hyperparameter Tuning Approach for Extreme Gradient Boosting." in *2019 Fifteenth International Conference on Information Processing (ICINPRO)*. Bangaluru, 2019.

¹⁶ L. Merrick and T. Ankur. "The explanation game: Explaining machine learning models using shapley values." *Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, 2020, Proceedings 4*. Springer International Publishing, 2020.