Prediction and Classification of Asteroid Orbit Classes using Machine Learning

Leen Almousa
Computer Engineering Department
Princess Sumaya University for Technology
Amman, Jordan
lee20210268@std.psut.edu.jo

Tala Hammouri
Computer Engineering Department
Princess Sumaya University for Technology
Amman, Jordan
tal20210061@std.psut.edu.jo

Lojain Hamadan
Computer Engineering Department
Princess Sumaya University for Technology
Amman, Jordan
loj20210576@psut.edu.jo

***Abstract*—Machine Learning is used in many fields of study. This paper used machine learning to classify asteroids from the Orbit Dataset for All Known Asteroids into their orbit type. Supervised learning was used to make the classification. Multiple machine learning models were built: Decision Trees, K-Nearest Neighbors(KNN), Multi-layer Perceptron, Naïve Bayes Classifier, and Random Forest. Multi-layer Perceptron performed the best with 100% accuracy. The worst-performing algorithm was the K-Nearest Neighborwith 83% accuracy.**

I. Introduction

Asteroids are rocky, tiny objects that revolve around the Sun [1]. An orbit is the consistent, repeating path one asteroid in space follows as it moves around another [2]. These orbital paths differ due to many factors, such as the asteroid's initial velocity and the gravitational force from planets, leading to various asteroid orbit classes that carry crucial insights into the early solar system [3].

The Minor Planet Center (MPC) is a key organization responsible for gathering the observations of asteroids, comets, and other small bodies in the Solar System. Funded by NASA's Near-Earth Object Observations program grant, the MPC created a comprehensive database containing hundreds of thousands of orbital object records, helping professional and beginner researchers study and classify asteroids.[4] This dataset is freely available to the public and follows open-source principles, encouraging more research in planetary science.

Classifying asteroid orbits is the process of categorizing them based on their shared characteristics. This classification is crucial to understand their origin, dynamics, and behavior to avoid any potential danger to Earth [5]. As traditional classifying ways often rely on time-consuming calculations, which becomes a struggle considering the growing number of asteroids and their complex orbits, scientists have turned to machine learning to better classify asteroid orbits. This paper applied multiple classification algorithms to perform multi-class classification to categorize asteroid orbits into distinct classes based on the MPC dataset.

This paper is divided into five segments. The first segment reviewed previously published work in the asteroid orbit classification field. The second segment described the MPC dataset, and the preprocessing steps applied to prepare and clean the data. The third segment detailed the classification algorithms used and their hyperparameters. The fourth segment presented the results of the model evaluations. Finally, a summary and conclusion of the results were provided in the fifth segment.

II. Related Work

Asteroid classification is no novel topic, and while there have been many studies conducted on the prediction of potentially hazardous asteroids (PHAs) which use different combinations of orbital parameters, including the orbit classes of the asteroids, little research has been published on the prediction of the asteroid's orbit class using different machine learning algorithms.

One paper focusing on asteroid classification, published by Tibrewal and Dwivedi [6], presented their findings on the application of the Radial Basis Function (RBF) kernel in the Support Vector Machines (SVM) algorithm. Their research underscores the significance of asteroid classification in advancing our understanding of orbital mechanics and potential hazard identification. The study systematically experimented with the parameter tuning of the RBF kernel to optimize SVM performance in this domain. The results demonstrated that the RBF SVM algorithm is highly effective, achieving an impressive accuracy rate of 97.9%.

Off-topic, but in the field, an Icarus-issued volume [7] on the classification of asteroids using G-mode analysis, separated into seven taxonomic units, with a confidence level of 99.7%. Another study on the classification of asteroid spectra yielded an average of 90% accuracy [8].

Different machine learning algorithms, namely Random Forest, Decision Tree, K Nearest Neighbors (KNN), Naive Bayes, Logistic Regression, and Support Vector Machine (SVM) were used for identifying potential hazardous and non-hazardous asteroids [9]. Sandeep and Reddy [10] proposed an early warning system for hazardous asteroids using advanced algorithms to predict, categorize, and assess the threat levels posed by asteroids and distinguish between two critical classes of asteroids: Near-Earth Objects (NEOs), which routinely cross Earth's orbital path, and potentially hazardous asteroids.

Another study used a supervised quantum machine learning approach, which leveraged the quantum properties of the data to improve the accuracy and precision of asteroid classification employed to detect hazardous asteroids based on their parameters [11]. Kumar's work [12] on the prediction of comets into orbit classes is aligned with our intended topic but on a different celestial body altogether.

Additionally, studies have used the Sloan Digital Sky Survey (SDSS) to classify objects into several stellar categories, including dwarfs, giants [13], and even galaxies and stars[14]. Machine learning methods were used to analyze and categorize these items based on their attributes.

III. Experimental Setup

The dataset used in this paper was collected by the Minor Planet Center and was then uploaded to Kaggle. This paper aims to classify these instances into the following categories: Main Belt Asteroid (MBA), Phocaea, Object with perihelion distance < 1.665 AU, Hilda, Amor, Hungaria, Jupiter Trojan, Apollo, Distant Object, Aten or Atira.

A. *Attribute Information*

The dataset contains 102,510 instances, distributed as shown in Table 1. It is clear that there is a huge imbalance in the dataset.

Table 1: Distribution of Asteroids by Orbit Type

Class	Number of instances
MBA	951,452
Hungaria	25,449
Phocaea	11,400
Jupiter Trojan	9,423
Object with perihelion distance < 1.665 AU	16,810
Hilda	4,643
Apollo	12,610
Amor	10,373
Distant Object	4,026
Aten	1,885
Atira	50

Some of the features had missing values; to avoid any inaccuracy, these instances were dropped. Furthermore, some of the features were not useful for determining which class the instance belonged to; hence, they were excluded. After excluding the irrelevant features, the dataset was reduced to twelve features and one label from the original thirty-four features and one label dataset. Table 2 shows the features used, their type and a brief description of them.

Table 2: List of Selected Features

Feature	Type	Description
H	float	The measurement of an object's brightness.
Num observations	float	The number of observations used to determine the orbit.
Perturbers_2	string	The count of secondary gravitational influences or celestial bodies is considered in the calculation of an object's orbit.
E (Eccentricity)	float	The parameter that describes the curvature of the orbit the asteroid has.
N	float	The distance the asteroid moves per day.
A	float	The semi-major axis of orbit.
Hex flags	string	Hexadecimal flags that indicate extra asteroid attributes.
Last observation	string	The date of the last observation.
Perihelion distance	float	The farthest distance of the asteroid from the sun.
Aphelion distance	float	The farthest distance of the asteroid from the sun.
Semilatus distance	float	The characteristic of the orbit follows: $\text{Semilatus distance} = a \cdot (1 - e^2)$ A: is the semi-major axis E: is eccentricity
Synodic period	float	The time it takes for an asteroid and Earth to align with the sun.

Before proceeding to data preprocessing, a histogram and boxplot of each attribute were plotted in Figures 1-20. It was clear that feature scaling would be necessary for more accurate predictions. The distribution of A, Perturbers_2, Aphelion_dist and Synodic period are shown in Figures 5, 6, 9,10,15, 16, 19 and 20, respectively might make it hard for the models to make predictions. Figure 1 follows a similar distribution to a normal distribution.

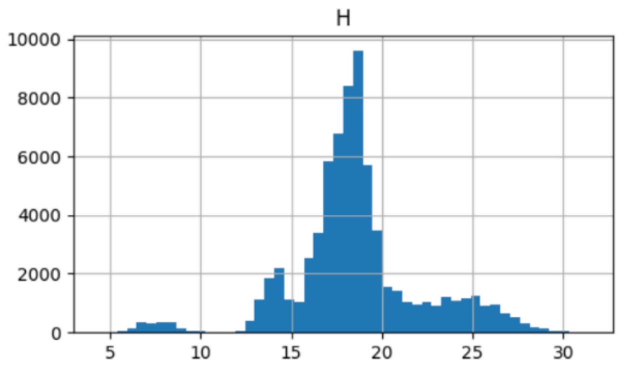


Figure 1:Histogram plot of the Feature H Preprocessing

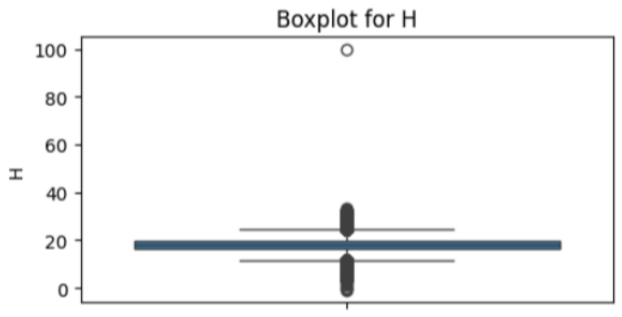


Figure 2: Boxplot of the Feature H Preprocessing

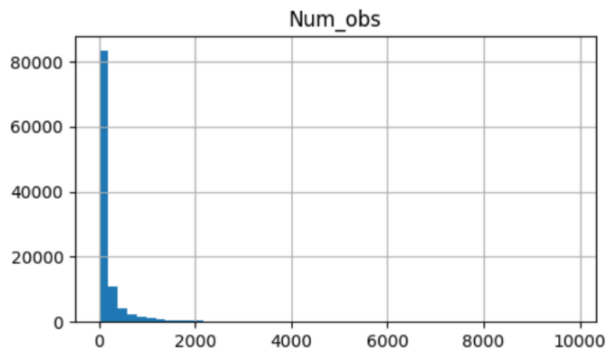


Figure 3:Histogram plot of the Feature Num_obs Preprocessing.

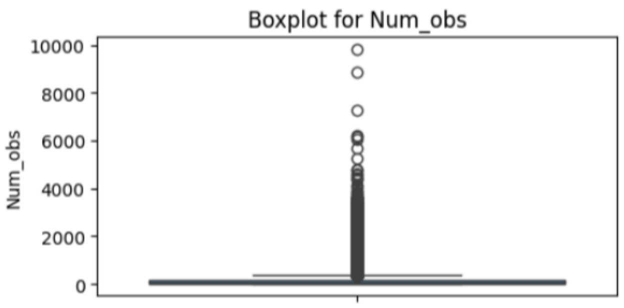


Figure 4:Boxplot of the Feature Num_obs Preprocessing.

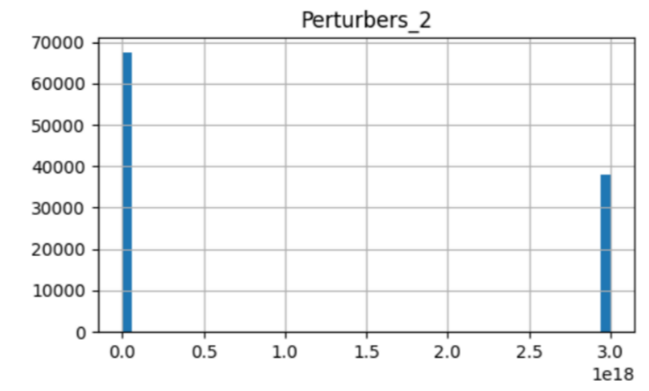


Figure 5:Histogram plot of the Feature Perturbers_2 Preprocessing

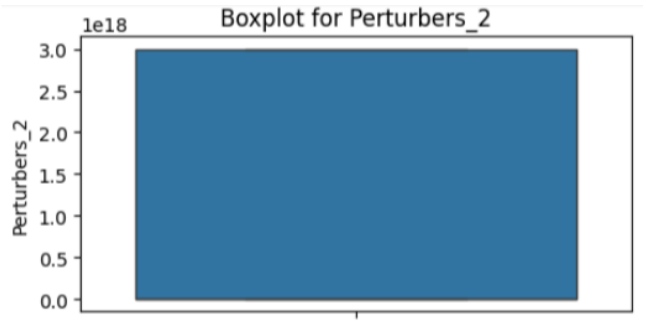


Figure 6:Boxplot of the Feature Perturbers_2 Preprocessing

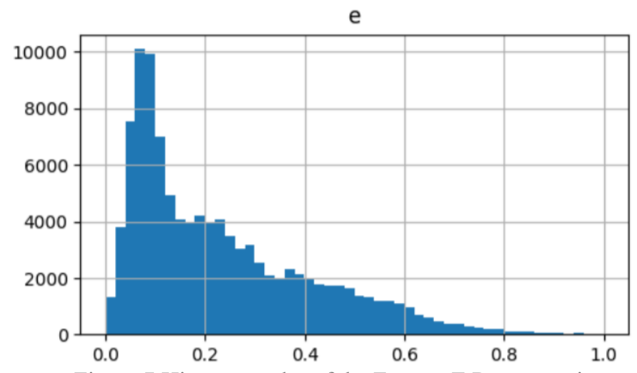


Figure 7:Histogram plot of the Feature E Preprocessing

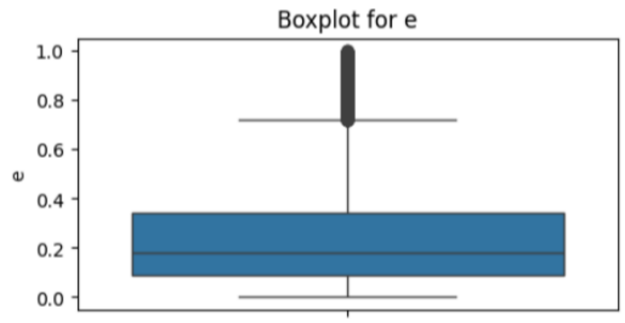


Figure 8:Boxplot of the Feature E Preprocessing

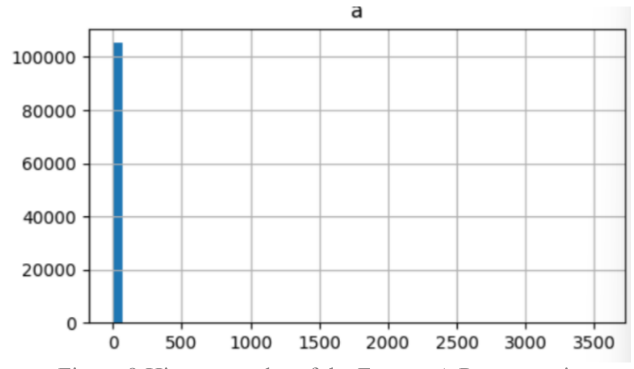


Figure 9:Histogram plot of the Feature A Preprocessing

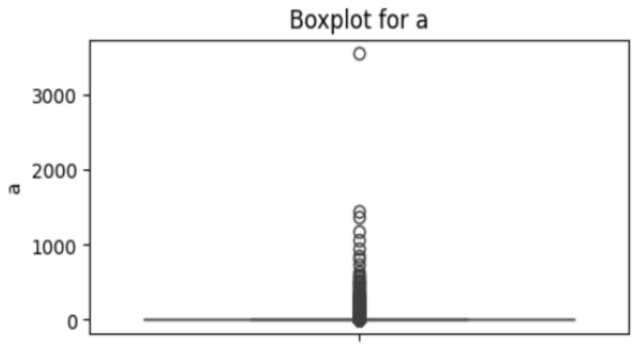


Figure 10:Boxplot of the Feature A Preprocessing

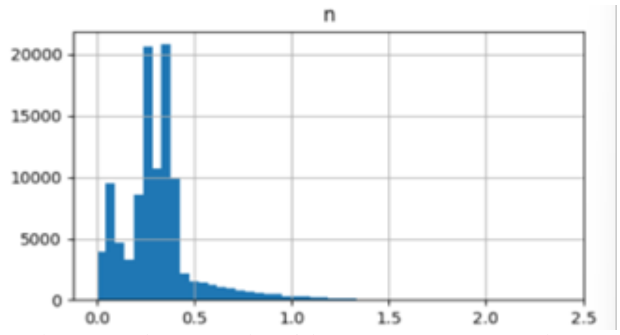


Figure 11:Histogram plot of the Feature N Preprocessing

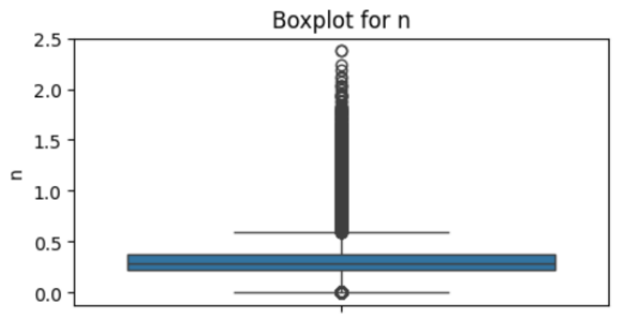


Figure 12:Boxplot of the Feature N Preprocessing

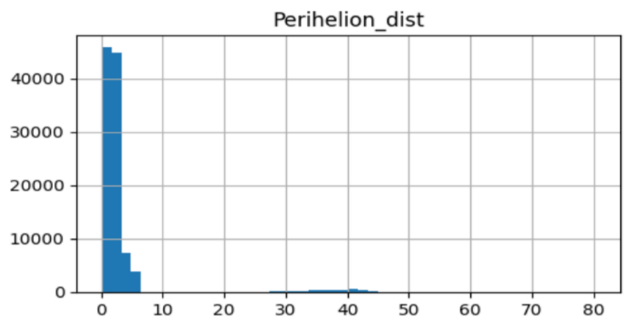


Figure 13:Histogram plot of the Feature Perihelion_dist Preprocessing.

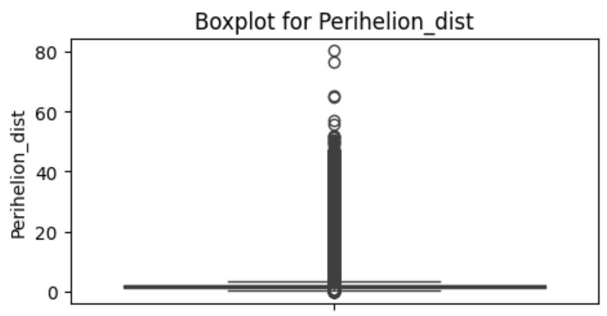


Figure 14:Boxplot of the Feature Perhelion_dist Preprocessing.

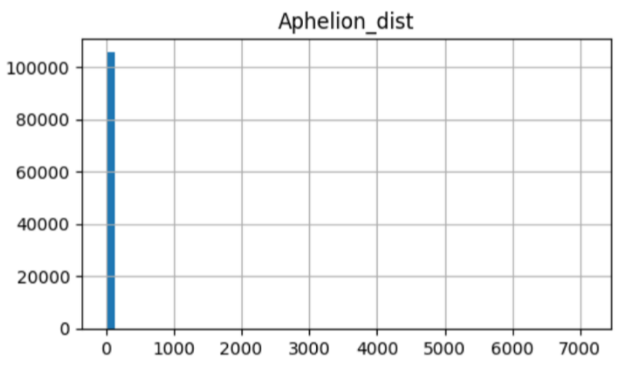


Figure 15:Histogram plot of the Feature Aphelion_dist Preprocessing.

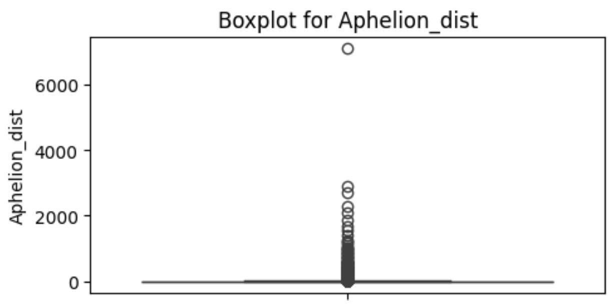


Figure 16:Boxplot of the Feature Aphelion_dist Preprocessing.

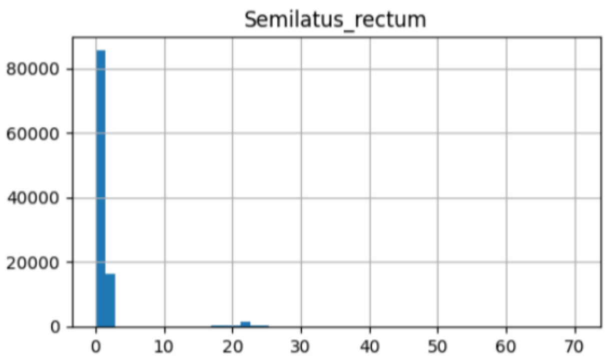


Figure 17:Histogram plot of the Feature Semilatus_rectum Preprocessing.

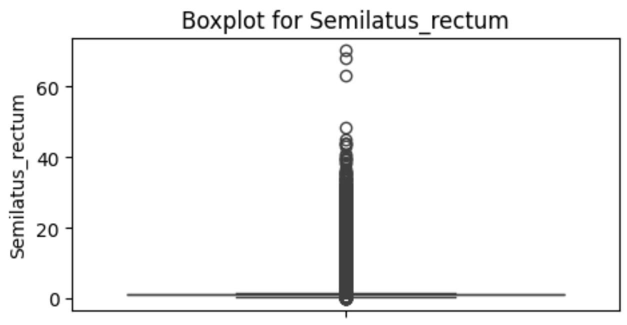


Figure 18:Boxplot of the Feature Semilatus_rectum Preprocessing.

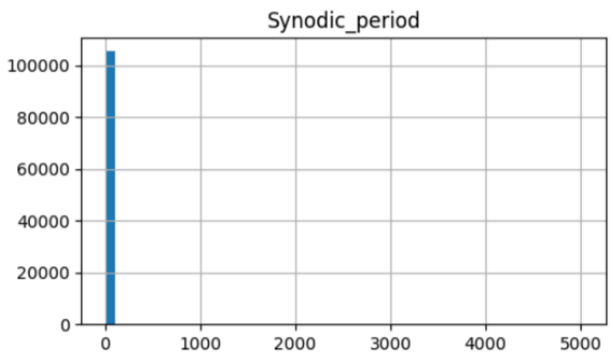


Figure 19:Histogram plot of the Feature Synodic_period Preprocessing.

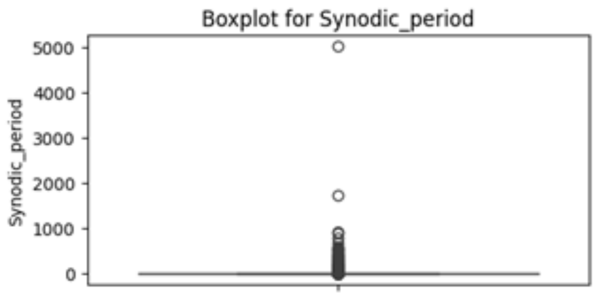


Figure 20:Boxplot of the Feature Synodic_period Preprocessing.

The correlation between these features has been plotted. The resulting correlation matrix is presented in Figure 21, showing the relation between these features. As seen there is a high correlation between Aphelion distance and feature a. Also, features Semilatus rectum and Perihelion distance demonstrated a high correlation.

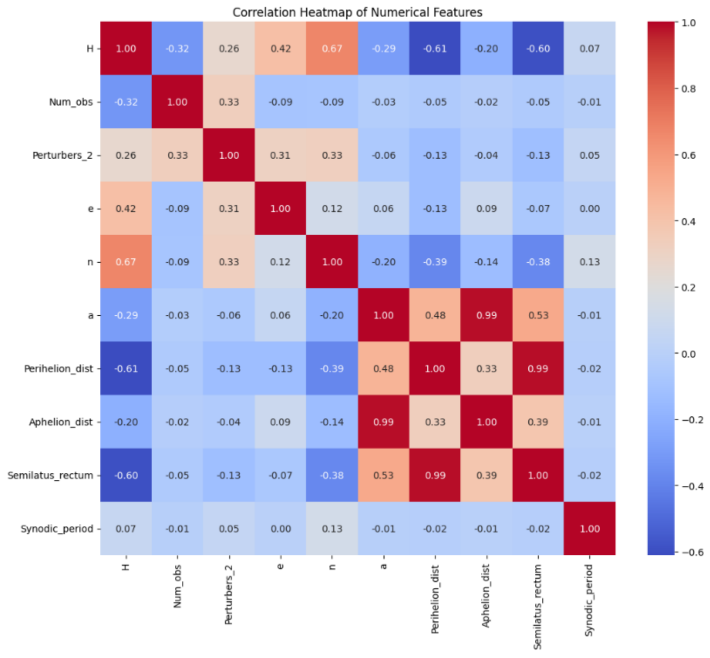


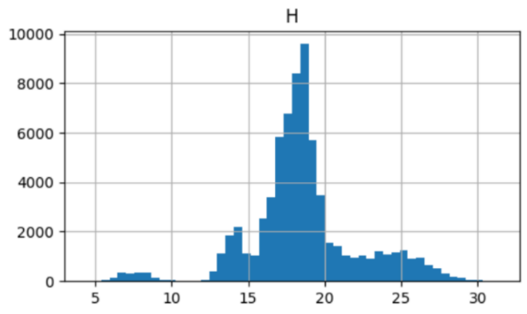
Figure 21:Heatmap of the Correlation between numerical features Preprocessing.

B. Data Preprocessing

Firstly, duplicated instances and instances containing null values were excluded to ensure data integrity. Given the significant class imbalance in the dataset, with the majority class being MBA, the MBA class was under-sampled to match the number of samples in the Hungaria class. This is an attempt to try and balance the dataset and reduce bias during model training.

After that, feature scaling was applied to normalize the dataset. Feature scaling is essential to avoid any bias towards any features since features differ in data ranges. Moreover, Gaussian Naïve Bayes performs better when the data is normally distributed, which ensures the need for scaling. To address skewness in the data, logarithmic scaling was applied, followed by standard scaling to standardize the feature values.

Next, the dataset was split into training and testing sets, by the following ratios respectively 80% and 20% which was necessary to see how well the model generalized to unseen data. The data was split randomly. The split was stratified based on the "Orbit_type" label to guarantee that the percentage of this feature was consistent across both training and testing datasets, ensuring more accurate and reliable results.



Figures 22-31 represent the training data after preprocessing; however, these techniques were applied to both training and testing data.

Figure 22: Histogram of the Feature H Postprocessing

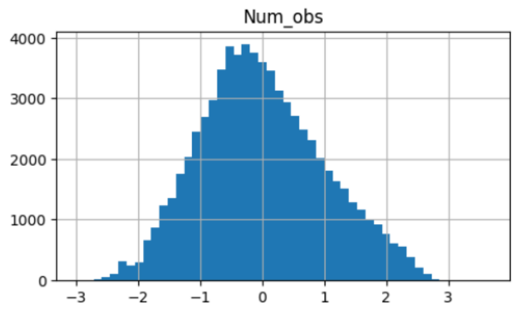


Figure 23: Histogram of the Feature Num_obs Postprocessing.

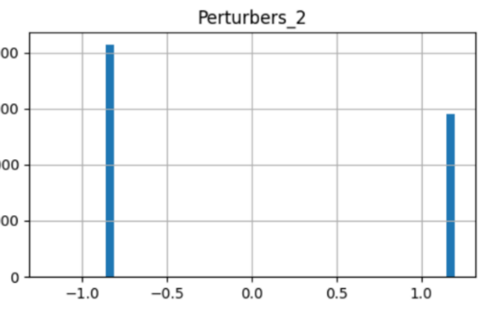


Figure 24:Histogram of the Feature Perturbers_2 Postprocessing

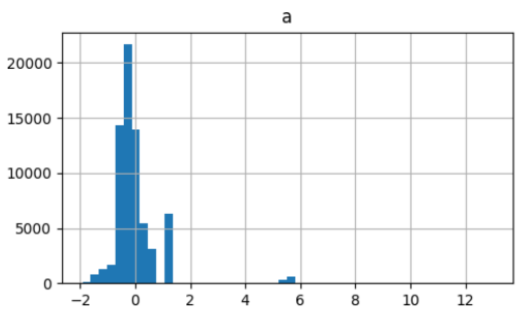


Figure 25:Histogram of the Feature A Postprocessing

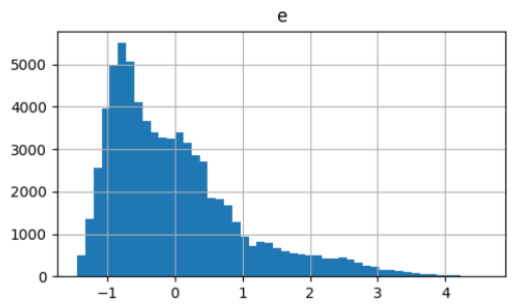


Figure 26:Histogram of the Feature E Postprocessing

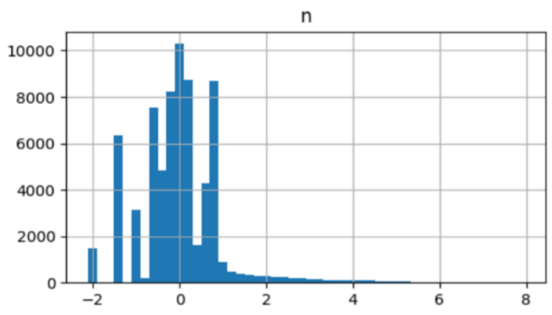


Figure 27: Histogram of the Feature N Postprocessing

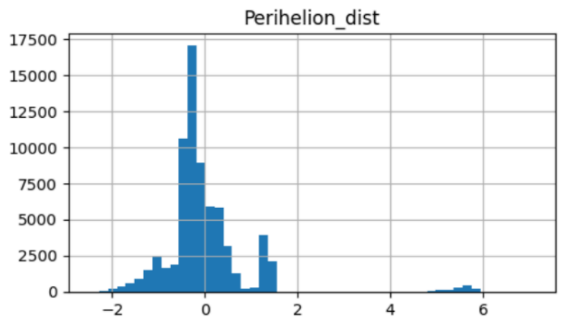


Figure 28: Histogram of the Feature Perihelion_dist Postprocessing.

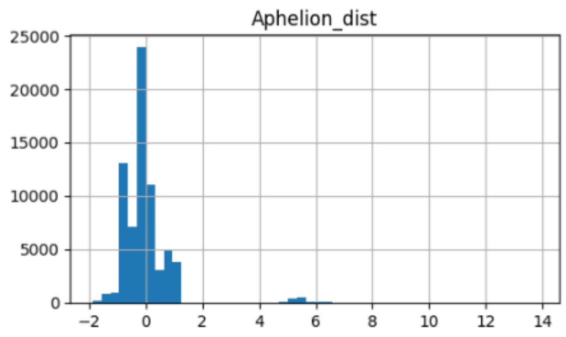


Figure 29: Histogram of the Feature Aphelion_dist Postprocessing.

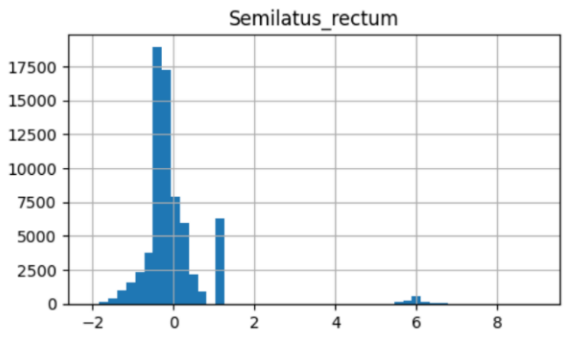


Figure 30: Histogram of the Feature Semilatus_rectum Postprocessing.

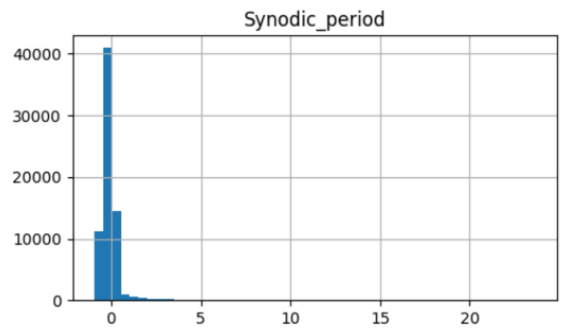


Figure 31: Histogram of the Feature Synodic_period Postprocessing

Further preprocessing was performed on the training dataset, which involved removing outliers specific to each class to prevent the removal of minority classes. Additionally, to solve the imbalance between the classes, the minority class, Atira, was oversampled to guarantee that the model had enough samples for training. These techniques helped to create a more balanced data set that was later used for model development. Figures 32-41 are boxplots representing each feature after the outliers were removed, this aims to visualize the new distribution of the data.

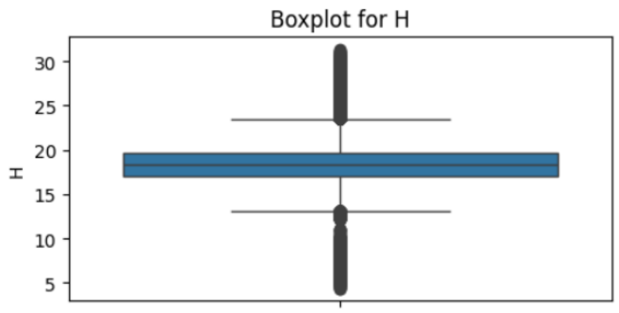


Figure 32:Boxplot of the Feature H Postprocessing

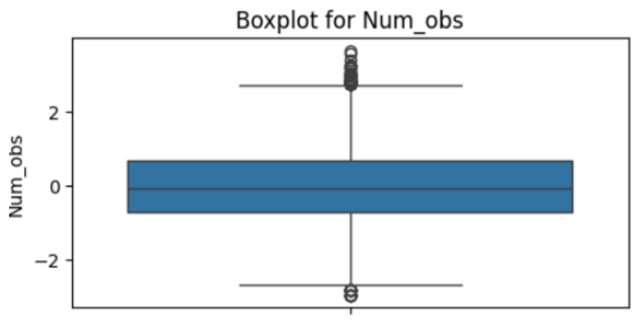


Figure 34:Boxplot of the Feature Num_obs Postprocessing.

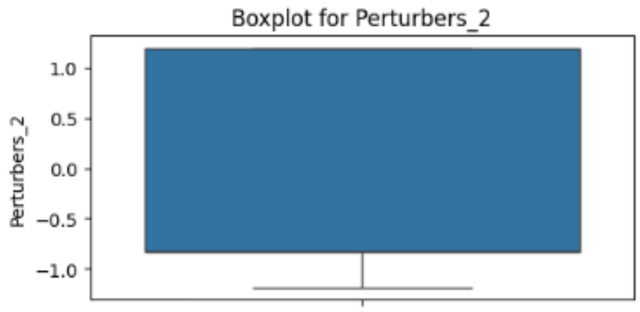


Figure 33:Boxplot of the Feature Perturbers_2 PostProcessing

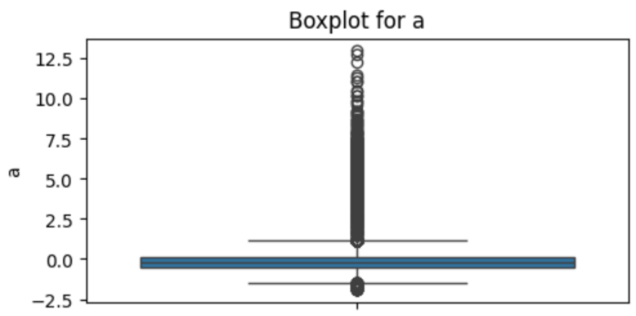


Figure 35: Boxplot of the Feature A Postprocessing

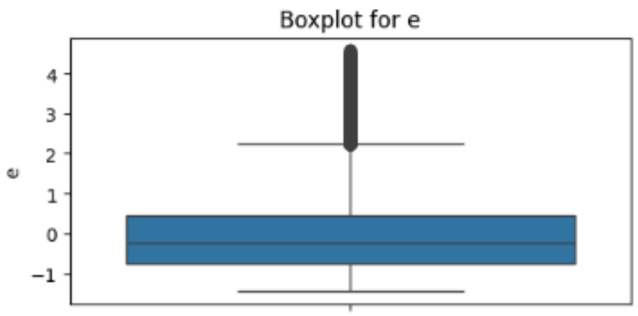


Figure 36: Boxplot of the Feature E Postprocessing

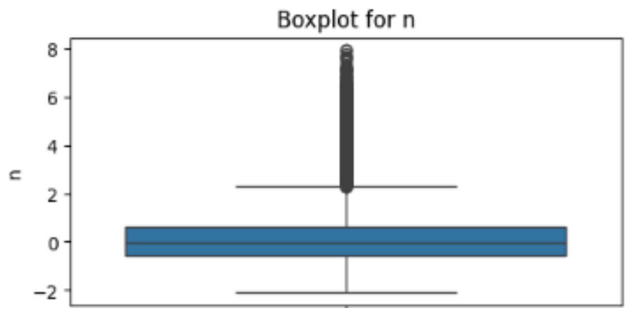


Figure 37: Boxplot of the Feature N Postprocessing

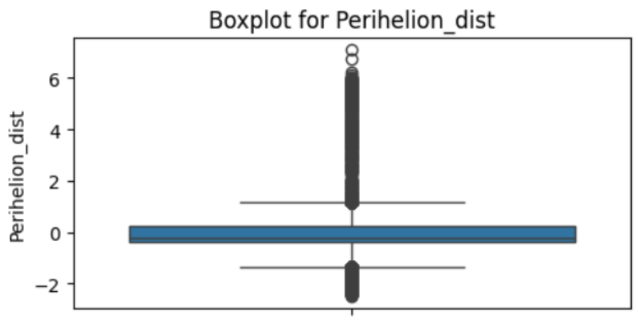


Figure 38: Boxplot of the Feature Perihelion_dist Postprocessing.

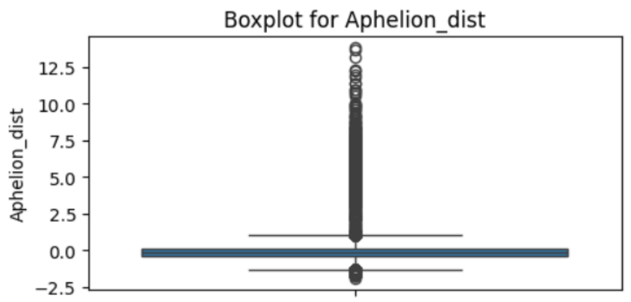


Figure 39:Boxplot of the Feature Aphelion_dist Postprocessing.

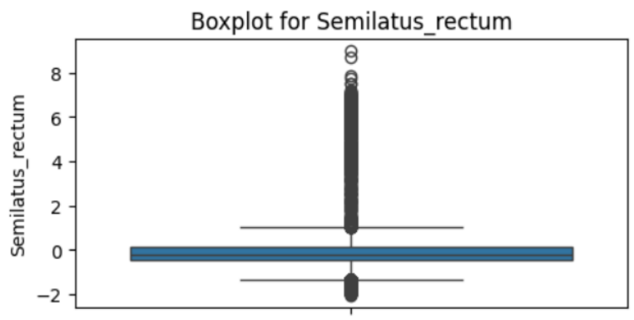


Figure 40: Boxplot of the Feature Semilatus_rectum Postprocessing.

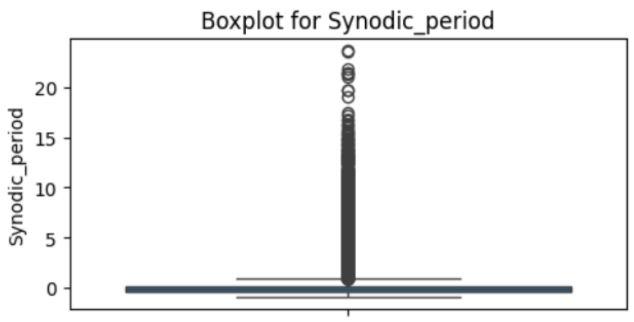


Figure 41: Boxplot of the Feature Synoic_period Postprocessing.

After scaling these features, Figure 42 was plotted to explore any correlations between the features. There was a strong correlation of 0.98 between features a Semilatus_rectum, a strong correlation of 0.98 between features a and Aphelion_dist, a strong correlation of 0.98 between features Semilatus_rectum and Perhelion_dist, a strong correlation of 0.94 between features a and Perhelion_dist, and a, and a strong correlation of 0.93 between features Aphelion_dist, and Semilatus_rectum.

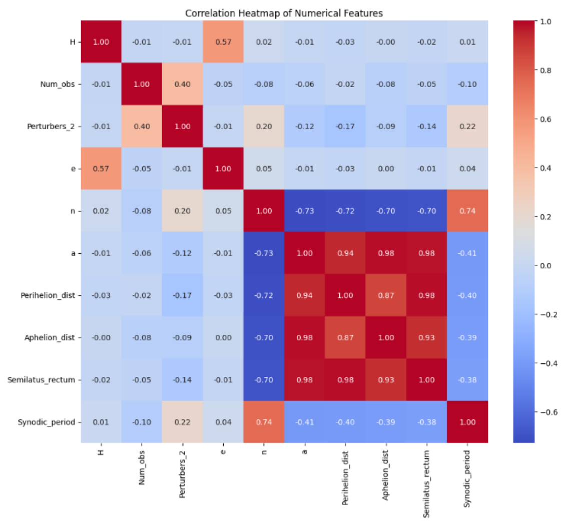


Figure 42:Heatmap representing the Correlation between Numerical Features Postprocessing

Before training the algorithms, Principal Component Analysis (PCA) was performed in an attempt to try reducing the number of features given to the model. However, it resulted in ten components which are equal to the number of features hence it was dropped.

Figure 43 represents the correlation between the numerical features and the label (Orbit type).

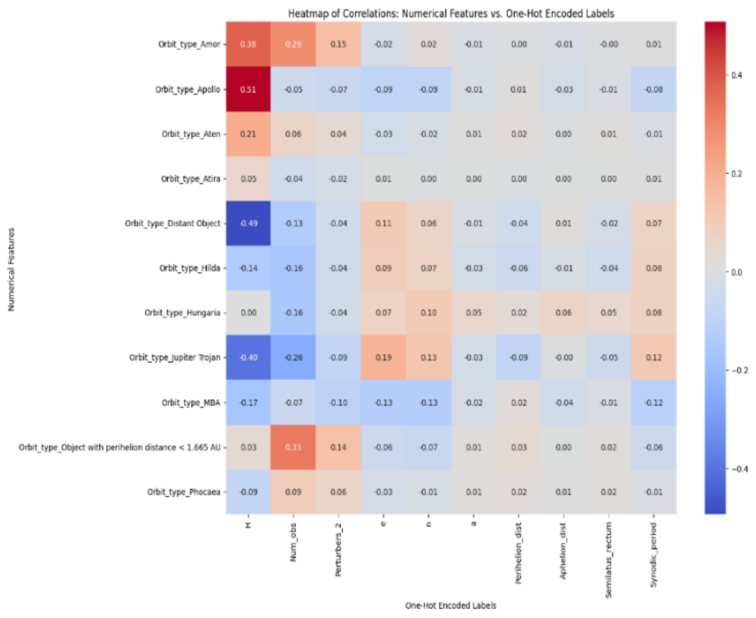


Figure 43:Heatmap representing the Correlation between Numeric Features and the Orbit Type Label

IV. ALGORITHMS

After analyzing and visualizing the data, the training set was and randomly shuffled and split into five folds for cross-validation. A select number of classifier models were fitted, and performance metrics determined the discrepancies between the models.

Hyperparameters of each respective model were tuned, to achieve optimal values and increase accuracy. In retrospect, using grid search for tuning was optimal; however, due to the limited processing capabilities relative to the dataset size, it was used for only the Random Forest classifier.

The working principle of the top-scoring classifier models and their hyperparameters are short-listed below:

A. Decision Tree

A decision tree is a supervised machine learning algorithm that follows a hierarchical structure formed by the rule-based partitioning of the training set, stemming from a root node down to the final verdict at the leaf node. Instance values at each node are tested against a set of rules, and outcomes represented by branches are traced to a unique path leading to the predicted class label.

Feature selection at each split relies on metrics such as entropy and Gini impurity as seen in Equation 1, which measures how frequently a randomly chosen element would be misclassified, defined as:

$$Gini = \sum_{i=1}^{C} {p_i}^2 \tag{1}$$

where p is the probability of class in a node, and C is the number of classes.

The algorithm's non-parametric nature and flexibility in handling both numeric and categorical features make it suitable for the problem set. Unfortunately, it is prone to overfitting and, in cases of highly imbalanced datasets, tends to be biased toward the larger sample. The hyperparameters used in the

Decision Tree model are shown in Table 3.

Table 3: Hyperparameters for the Decision Tree Model.

Hyperparameter	Value
Min Sample Leaf	1
Max Samples Split	2
Criterion	**Gini**

B. Random Forest

Random Forest is an ensemble of decision trees, a popular supervised learning method for classification and regression. Random Forests use bagging, a technique that takes random subsets (bootstrap samples) of the training data and builds parallel decision trees. This randomness results in a low correlation between individual trees, thus reducing the risk of bias.
Hard voting is implemented, where each decision tree predicts a class label, and the final prediction is the class with the majority of votes from all the trees. The aggregation of a larger number of trees results in less noise in the training data, which makes it less prone to overfitting.
The best hyperparameter values found over two iterations using random search are shown in Table 4.

Table 4: Hyperparameters for the Random Forest Model.

Hyperparameter	Value
Max Depth	30
Max Samples Split	13
Estimators	107

C. K-Nearest Neighbors

K-Nearest Neighbors (KNN) is an instance-based, proximity-based supervised learning method that is widely used for both classification (with discrete labels) and regression (with continuous labels).

Prediction of new instances is based on the majority vote of the classes of the K neighbors relative to the input query point, partitioning them into different regions.

The value of k, the number of neighbors to consider, is a balancing factor between overfitting and underfitting. Large values of k may result in high bias and low variance, and lower values may lead to the opposite. K is usually an odd number to avoid ties.

This algorithm classifies an instance based on the majority class of the K-nearest neighbors. An odd value for K is typically used to avoid a tie between classes. Hyperparameter tuning was also done to increase the accuracy, these hyperparameters are shown in Table 5.

The proximity of data points is determined using distance metrics. The chosen metric used by the model is the Minkowski distance as shown in Equation 2.

$$Minkowski\ Distance = \sqrt{\sum_{i=1}^{n} |x_i - y_i|}^{\frac{1}{p}} \qquad (2)$$

Other metrics, such as Euclidean Distance or Manhattan Distance, may also be used depending on the problem.

Table 5: Hyperparameters for the K-Nearest Neighbors Model.

Hyperparameter	Value
N Neighbours	5
Weights	Uniform
Power Parameter	2
Leaf Size	30

D. Multi-layer Perceptron

The next supervised learning model trained was Multi-layer Perceptron, an artificial neural network. The structure of the model is as follows: one input layer, one or more hidden layers, and an output layer. Every neuron in all layers relates to every neuron in the next layer, except for the output layer.

All neurons except for input neurons are passed through an activation function. The output of an activation function decides whether the neuron should fire or not. The activation function used in this model was the Rectified Linear Unit (ReLu) function as shown in Equation 3.

$$f(z) = \max(0, z) \qquad (3)$$

Multi-layer Perceptrons have many hyperparameters that can be optimized. Three of the chosen hyperparameter values are shown Table 6.

Table 6: Hyperparameters for the Multi-layer Perceptron Model.

Hyperparameter	Value
Hidden Layer Sizes	(100,)
Alpha	0.0001
Max Function	15,000

E. Naïve Bayes Classifier

The fifth algorithm used was Gaussian Naïve Bayes, a probabilistic classifier; it uses Bayes theorem of probability to predict the class of instances, which is defined in Equation 4.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \qquad (4)$$

The algorithm assumes independence among features. The Gaussian Naïve Bayes algorithm also assumes that the features follow a normal distribution. The chosen hyperparameter for smoothing is shown in Table 7.

Table 7: Hyperparameters for the was Gaussian Naïve Bayes Model.

Hyperparameter	Value
Var Smoothing	1e-09

V. Results and Analysis

This section demonstrates the results and highlights discrepancies between the tested algorithms. The classification report included metrics such as the accuracy across different folds and their mean, precision, recall, F1 scores, and the confusion matrix generated for each model. Limited performance capabilities deterred testing more algorithms.

Accuracy is the percentage of correct predictions, whether positive or negative, over the entire dataset and is calculated using Equation 5.

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \tag{5}$$

where TN = True Negatives TP = True Positive
FN =False Negative FP = False Positives

Precision measures the percentage of correct positive predictions among all positive predictions, as seen in Equation 6.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

Recall measures the percentage of correctly classified actual positives overall positives, it is calculated using Equation 7.

$$Recall = \frac{TP}{TN + FP + TP + FN} \tag{7}$$

F1-score is the derived harmonic mean of precision and recall, useful when there is a class imbalance as it penalizes extreme negatives. It is calculated using Equation 8.

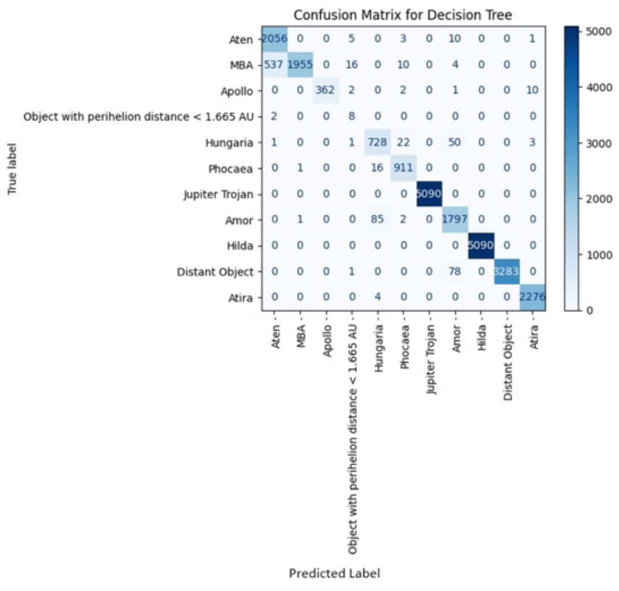
$$F1\ Score = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \tag{8}$$

The Decision Tree's classification report shown in Table 8 revealed high scores across all metrics for most classes, notably achieving perfect scores for classes such as "Hungaria," "MBA,"and "Phocaea."However, the algorithm struggled with the "Atira" class, showing a significantly lower precision (0.24) and F1-score (0.37), due to its small sample size (support = 10). Despite this, the overall accuracy of the model is 96%.

Table 8: Classification Performance Metrics by Class for the Decision Tree Model.

Class	Precision	Recall	F1-Score	Support
Amor	79%	99%	88%	2075
Apollo	100%	78%	87%	2522
Aten	100%	96%	98%	377
Atira	24%	80%	37%	10
Distant Object	87%	90%	89%	805
Hilda	96%	98%	97%	928
Hungaria	100%	100%	100%	5090
Jupiter Trojan	93%	95%	94%	1885
MBA	100%	100%	100%	5090
Object with perihelion distance < 1.665 AU	99%	100%	100%	3362
Phocaea	99%	100%	100%	2280
Accuracy		96%		24424
Macro Average	89%	94%	90%	24424
Weighted Average	97%	96%	96%	24424

The confusion matrix for the model is shown in Figure 44.



	Aten	MBA	Apollo	Object with perihelion distance < 1.665 AU	Hungaria	Phocaea	Jupiter Trojan	Amor	Hilda	Distant Object	Atira
Aten	2056	0	0	5	0	3	0	10	0	0	1
MBA	537	1955	0	16	0	10	0	4	0	0	0
Apollo	0	0	362	2	0	2	0	1	0	0	10
Object with perihelion distance < 1.665 AU	2	0	0	8	0	0	0	0	0	0	0
Hungaria	1	0	0	1	728	22	0	50	0	0	3
Phocaea	0	1	0	0	16	911	0	0	0	0	0
Jupiter Trojan	0	0	0	0	0	0	5090	0	0	0	0
Amor	0	1	0	0	85	2	0	1797	0	0	0
Hilda	0	0	0	0	0	0	0	0	5090	0	0
Distant Object	0	0	0	1	0	0	0	78	0	3283	0
Atira	0	0	0	0	4	0	0	0	0	0	2276

Figure 44:Confusion Matrix for the Decision Tree Model Results

Random Forest achieved an overall accuracy of 97% with a weighted average precision, recall, and F1-score of 97%, 97%, and 96%, respectively. Its Classification report is shown in Table 9. It excelled in correctly identifying frequent classes such as "Hungaria," "MBA," and "Object with perihelion distance < 1.665 AU," achieving near-perfect or perfect scores across all metrics. However, the model struggled with the "Atira" class, attaining a precision of 100% but a recall of only 30%, reflecting the challenges posed by its small sample size.

Table 9: Classification Performance Metrics by Class for the Random Forest Model.

Class	Precision	Recall	F1-Score	Support
Amor	97%	97%	97%	2075
Apollo	100%	81%	93%	2522
Aten	100%	87%	93%	377
Atira	100%	30%	46%	10
Distant Object	100%	99%	100%	805
Hilda	100%	100%	100%	928
Hungaria	90%	100%	95%	5090
Jupiter Trojan	100%	100%	100%	1885
MBA	96%	100%	98%	5090
Object with perihelion distance < 1.665 AU	100%	99%	100%	3362
Phocaea	100%	91%	95%	2280
Accuracy		97%		24424
Macro Average	98%	89%	92%	24424
Weighted Average	97%	97%	96%	24424

The algorithm's confusion matrix is shown in Figure 45.

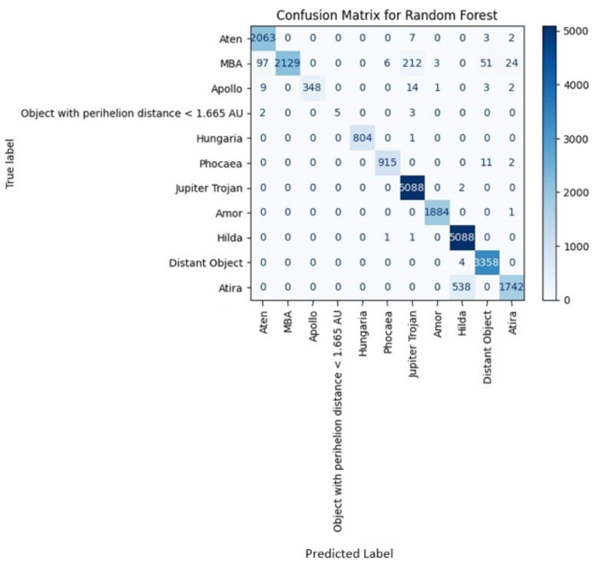


Figure 45:Confusion Matrix for the Random Forest Model Results

The K-Nearest Neighbors (KNN) classifier achieved an accuracy of 83%, the lowest among all evaluated algorithms. Its classification report is shown in Table 10. While it performed well for frequent classes such as "Hungaria" and "Object with perihelion distance < 1.665 AU," achieving F1-scores of 91% and 97%, respectively, it struggled significantly with rarer classes like "Atira" and "Aten," where F1-scores dropped to 3% and 9%.

Table 10: Classification Performance Metrics by Class for the K-Nearest Neighbors Model.

Class	Precision	Recall	F1-Score	Support
Amor	52%	90%	66%	2075
Apollo	87%	17%	28%	2522
Aten	77%	5%	9%	377
Atira	2%	20%	3%	10
Distant Object	92%	90%	91%	805
Hilda	94%	70%	80%	928
Hungaria	88%	95%	91%	5090
Jupiter Trojan	95%	90%	92%	1885
MBA	85%	100%	92%	5090
Object with perihelion distance < 1.665 AU	96%	97%	97%	3362
Phocaea	88%	80%	84%	2280
Accuracy		83%		24424
Macro Average	78%	68%	83%	24424
Weighted Average	86%	83%	81%	24424

Figure 46 illustrates the confusion matrix of the K-Nearest Neighbors (KNN) algorithm.

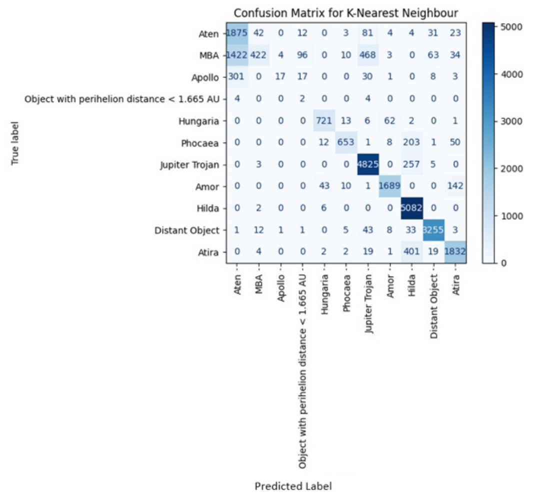


Figure 46:Confusion Matrix for the K Nearest Neighbor Model Results

The Naive Bayes classifier achieved an overall accuracy of 95% across 24,424 samples. Its results are demonstrated in Table 11. The performance varies across classes, with some categories like "Hungaria" and "MBA" achieving high F1-scores (99% and 98%, respectively), while others such as "Atira" and "Aten," performed lower, with F1-scores of 45% and 69%. The weighted average F1-score of 96% matches the overall accuracy, which means the model does an excellent job with common classes but struggles with less common ones like "Atira".

Table 11: Classification Performance Metrics by Class for the Naive Bayes Model.

Class	Precision	Recall	F1-Score	Support
Amor	100%	97%	95%	2075
Apollo	100%	91%	95%	2522
Aten	60%	82%	69%	377
Atira	30%	90%	45%	10
Distant Object	79%	93%	86%	805
Hilda	84%	97%	90%	928
Hungaria	99%	98%	99%	5090
Jupiter Trojan	88%	100%	94%	1885
MBA	99%	98%	98%	5090
Object with perihelion distance < 1.665 AU	99%	94%	97%	3362
Phocaea	95%	96%	96%	2280
Accuracy		95%		24424
Macro Average	85%	94%	88%	24424
Weighted Average	96%	96%	96%	24424

The Naïve Bayes confusion matrix is presented in Figure 47.

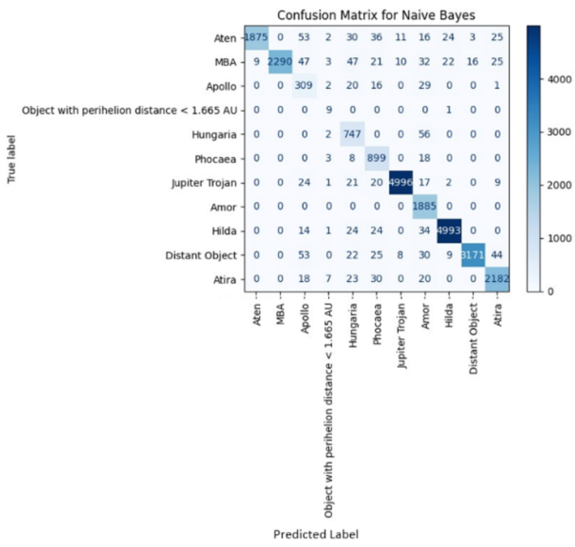


Figure 47: Confusion Matrix for the Multi-Layer Perceptron Model Results

The Multi-layer Perceptron achieved an accuracy of 100%, the highest among all algorithms. It performed the same as other models for frequent classes like "Hungaria" and "MBA," with both precision and recall at 100%. For the rarer classes like "Atira," it achieved a high recall of 80% and precision of 73%. It also maintained 100 % precision and recall scores in categories such as "Amor," "Apollo," and "Distant Object," The Multi-layer Perceptron demonstrated great consistency across all other classes, making it the best-performing algorithm overall. Its report is presented in Table 12.

Table 12: Classification Performance Metrics by Class for the Multi-Layer Perceptron Model.

Class	Precision	Recall	F1-Score	Support
Amor	99%	100%	100%	2075
Apollo	100%	99%	99%	2522
Aten	99%	99%	99%	377
Atira	73%	80%	76%	10
Distant Object	100%	100%	100%	805
Hilda	100%	100%	100%	928
Hungaria	100%	100%	100%	5090
Jupiter Trojan	100%	100%	100%	1885
MBA	100%	100%	100%	5090
Object with perihelion distance < 1.665 AU	100%	100%	100%	3362
Phocaea	100%	100%	100%	2280
Accuracy		100%		24424
Macro Average	97%	98%	98%	24424
Weighted Average	100%	100%	100%	24424

Figure 48 below demonstrates the performance of the Multi-layer Perceptron classifier:

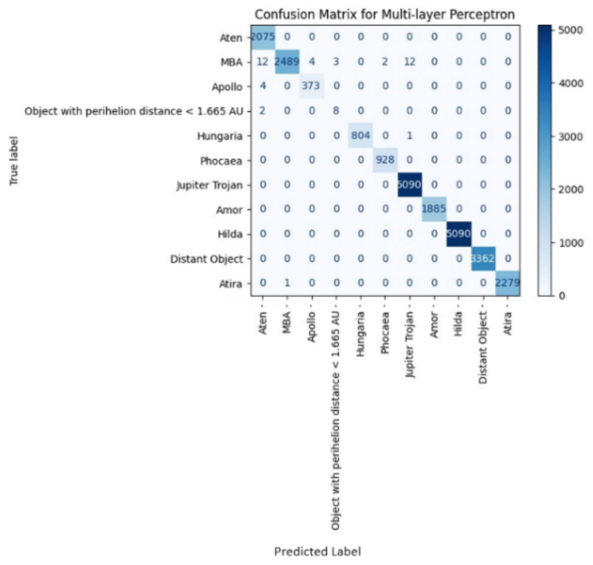


Figure 48:Confusion Matrix for the Multi-Layer Perceptron Model Results

In Table 13 a summary of the accuracy achieved by each classification algorithm was presented.

Table 13: Classification Performance Accuracy for all implemented models.

Algorithm	Accuracy
Decision Trees	96%
Random Forest	97%
Multi-layer Perceptron	100%
Naïve Bayes	95%
K- Nearest Neighbor	83%

When comparing our results with the previous results of people who worked on the same data set we can see that we have made a significant improvement, as we improved the accuracy from 97.9 % to 100%as demonstrated in Table 14.This mainly due to using more powerful algorithms such as Multi-layer perceptron whilst they used the SVM algorithm .

Table 14: Comparison between this Paper and Previous work

	Accuracy	Model Type
This paper	100%	Multi-Layer Perceptron
Previous work [1]	97.9%	SVM(RBF)

IV. Conclusion

This paper demonstrated the use of supervised learning to solve a multi-class classification problem based on the MPC database set. Five classification algorithms the Decision Trees, K-Nearest Neighbors, Multi-layer Perceptron, Naïve Bayes, and Random Forest were implemented and evaluated on the data. The Multi-layer Perceptron achieved the highest performance with 100% accuracy, the K-nearest neighbor had the lowest accuracy at 83%. All the algorithms performed the best with well-represented classes like "Hungaria" and "MBA", but their performance decreased with lower-represented classes such as "Atira" Adding more instances from underrepresented classes could significantly improve classification accuracy.

References

[1] Andreić, Željko. (2015). Asteroids. Rudarsko-Geološko-Naftni Zbornik. 31. 69-85. 10.17794/rgn.2016.1.5.

[2] "What Is an Orbit?," NASA Science Space Place, [Online]. Available: https://spaceplace.nasa.gov/orbits/en/

[3] National Geographic "orbits" [Online] Available: https://education.nationalgeographic.org/resource/orbit/

[4] Rudenko, Michael. (2015). Minor Planet Center: Data processing challenges. Proceedings of the International Astronomical Union. 10. 265-269. 10.1017/S174392131500839X.

[5] Pars LA. The classification of orbits. Mathematical Proceedings of the Cambridge Philosophical Society. 1929;25(1):1-19. doi:10.1017/S0305004100018491

[6] Tibrewal, Y., & Dwivedi, N. (n.d.). Orbit Classification of Asteroids Using Implementation of Radial Basis Function on Support Vector Machines. NPS International School, Singapore; SVKM's NMIMS Mukesh Patel School of Technology Management & Engineering,Mumbai,India. (https://ar5iv.org/html/2306.17138#bib.bib2)

[7] M.Antonietta Barucci, M.Teresa Capria, Angioletta Coradini, Marcello Fulchignoni, Classification of asteroids using G-mode analysis, Icarus, Volume 72, Issue 2,1987, Pages 304-324, ISSN 0019-1035, https://doi.org/10.1016/0019-1035(87)90177-1. (https://www.sciencedirect.com/science/article/pii/0019103587901771)

[8] Howell, E. S., E. Merényi, and L. A. Lebofsky (1994), classification of asteroid spectra using a neural network, *J. Geophys. Res.*, 99(E5), 10847–10865, doi:10.1029/93JE03575.

[9] B. R. M, A. G, A. J and N. J. K, "Hazardous Asteroid Prediction Using Machine Learning," 2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN), Vellore, India, 2023, pp. 1-6.

[10] A. Sandeep, P. J. Reddy and T. M. Perkin, "Planetary Defense Using Machine Learning Early Warning Systems for Hazardous Asteroids," 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS), Chennai, India, 2024, pp. 1-6.

[11] R. Bhavsar et al., "Classification of Potentially Hazardous Asteroids Using Supervised Quantum Machine Learning," in IEEE Access, vol. 11, pp. 75829-75848, 2023.

[12] A. M. Kumar, "Prediction of Comets into Orbit Classes Using Machine Learning", International Journal of Artificial Intelligence and Soft Computing (IJAISC) Vol.1, No.5, October 2022.

[13] Swathi, S., Saranya, S., Vijayalakshmi, K. and Aswini, V., 2024, June. Stellar classification using supervised machine learning. In *AIP Conference Proceedings* (Vol. 3122, No. 1). AIP Publishing.T. Chuntama, P. Techa-Angkoon and C. Suwannajak, "Multi-class Classification of Astronomical Objects in the Galaxy M81 using Machine Learning Techniques," in *2020 24th International Computer Science and Engineering Conference (ICSEC)*, 2020.

[14] D. Omat, J. Otey and A. Al-Mousa, "Stellar Objects Classification Using Supervised Machine Learning Techniques," 2022 International Arab Conference on Information Technology (ACIT), Abu Dhabi, United Arab Emirates, 2022, pp. 1-8, doi: 10.1109/ACIT57182.2022.9994215. keywords: {Machine learning algorithms;Supervised learning;Stars;Support vector machine classification;Multilayer perceptrons;Classification algorithms;Decision trees;Machine Learning;Stellar Classification;Sloan Digital Sky Survey;Supervised Learning;Classificatio