# Outperforming State of The Art Asteroid Classification Using Simple Random Forest

**Ben Hassner***, **Michael Lemucchi**†, **Noor-Aysha Saadat**‡, **Nelsen Young**§
*3rd Year Computer Science Major, Project Lead
†3rd Year Computer Science Major, Economics Minor, Tech Associate
‡3rd Year Computer Science and Engineering Major, Tech Associate
§3rd Year Computer Science and Engineering Major, Tech Associate

**Abstract**

The potential threat posed by hazardous asteroids necessitates accurate and efficient detection methods to safeguard Earth. This paper aims to find the best application of machine learning algorithms to classify asteroids as hazardous or non-hazardous based on their features. The introductory section underscores the dangers of asteroid impacts and elaborates on critical asteroid features that determine their hazard potential. A comprehensive literature review follows, detailing various classification algorithms such as Logistic Regression, K-Nearest Neighbors (KNN), Decision Tree Classifier (DTC), and Random Forest. Additionally, previous work employing these algorithms in the context of asteroid classification is examined. In our research, we utilized a dataset from NASA comprising numerous asteroid features. After cleaning the dataset, we identified the most pertinent features: absolute magnitude, eccentricity, minimum orbit intersection, orbit uncertainty, perihelion distance, and relative velocity in km/hr. Subsequently, we optimized the dataset by determining the best random state for the train-test shuffle. We then evaluated several classifiers. Logistic Regression, with no hyperparameter tuning, achieved an accuracy of 92.6%. The KNN classifier, optimized for the best value of k-neighbors, reached an accuracy of 95.6%. The Decision Tree Classifier, with hyperparameter tuning for depth, achieved an accuracy of 99.7%. Lastly, the Random Forest classifier, fine-tuned for both max depth and the number of estimators, attained the highest accuracy of 99.8%. Our findings indicate that machine learning algorithms, particularly Random Forest, hold significant promise for the accurate classification of potentially hazardous asteroids, thereby contributing to the development of robust planetary defense strategies.

## I. Introduction and background

In the vastness of the outer Solar System, a single Oort Cloud asteroid floating aimlessly may appear miniscule, distant, and harmless. In reality, these extraterrestrial behemoths can reach diameters of up to 100 kilometers, equivalent to the size of the state of Rhode Island at its widest point (Salotti, 2022). If one of these colossal bodies were to impact Earth, the consequences would be catastrophic, rendering the planet inhospitable (Salotti, 2022). The kinetic energy released from such an impact would exceed one hundred billion megatons of energy: To put this into perspective, the largest man-made explosion ever was the former Soviet Union's Tsar Bomba, which only released 50 megatons of energy (Tikkanen, 2020). A single asteroid impact would release enough energy to boil the planet's oceans, eradicate all habitable environments, and completely erase all life forms from the face of the Earth (Salotti, 2022). Following an asteroid collision, submicrometer dust particles would permeate the atmosphere for decades, darkening the Earth's surface and halting photosynthesis. This dust layer would also cause temperatures to plummet, potentially triggering another ice age as the heat from the impact dissipates into space over time.

There is currently no reliable method to divert the course of a giant asteroid if one were to threaten Earth (Salotti, 2022). Despite ample warning time, current technologies are inadequate for destroying or predictably altering the trajectory of such an asteroid. To effectively protect our planet, it is imperative to thoroughly understand the threat posed by asteroids. The following sections will examine critical characteristics that influence an asteroid's potential danger. By scrutinizing these features in detail, we can develop a comprehensive understanding of the risks and enhance our strategies for planetary defense.

Certain classes of asteroids, such as Centaurs, exhibit notably unstable orbits, resulting in relatively short lifetimes. These asteroids often transition into regions like the Scattered Disk, a distant part of the outer solar system populated by small icy bodies known as Scattered Disk Objects (SDOs). These SDOs have orbits characterized by high eccentricity, meaning their paths are highly elongated and deviate significantly from perfect circles around the sun (Liu et al, 2023). As SDOs evolve, they can become Centaurs, whose unpredictable trajectories further increase the likelihood of entering the inner solar system. This orbital uncertainty raises the potential for these asteroids to impact Earth, underscoring the importance of closely monitoring their movements and characteristics (Salotti, 2022).

The perihelion distance refers to the closest point of an object's orbit around the Sun. For near-Earth asteroids (NEAs), this distance is an essential parameter that determines their proximity to Earth and potential hazard (Jin et al, 2022). NEAs are classified into four types based on their orbital characteristics: Amor-type, Apollo-type, Aten-type, and Atira-type. They are defined as asteroids with an orbital perihelion within 1.3 astronomical units (au) of the Sun. Among these, those with a perihelion distance within 0.05 au of Earth and an absolute magnitude brighter than 22 (indicating a diameter of about 140 meters) are designated as Potentially Hazardous Asteroids (PHAs) (Jin et al, 2022). For instance, Apollo-type and Aten-type NEAs have perihelion distances that bring

them close to or intersect with Earth's orbit, increasing their likelihood of posing a hazard. The minimum orbit intersection distance (MOI) refers to the closest point an asteroid's orbit comes to Earth's orbit, and it is a key parameter in determining the likelihood of an impact. NEAs with orbits that intersect or come close to intersecting Earth's orbit have a higher potential for posing a hazard (Jin et al, 2022). A smaller perihelion distance results in a smaller MOI, making the asteroid more dangerous. Conversely, Amor-type and Atira-type NEAs have perihelion distances that keep them outside and inside Earth's orbit, respectively, reducing their likelihood of impacting Earth. Thus, understanding the perihelion distance and MOI is essential for accurately assessing the impact risk posed by different types of near-Earth asteroids.

Exploring an asteroid's brightness, or magnitude, as observed from unveils another layer of understanding in asteroid research. The apparent magnitude of an object, such as an asteroid, specifically measures this brightness from Earth's vantage point (Liu et al, 2023). This measure is influenced by various factors, including the asteroid's distance from Earth, its size, and its reflective properties. Apparent magnitude can fluctuate due to changes in the object's distance from the Sun, variations in its phase angle, and atmospheric conditions such as clouds or turbulence during observations. Most asteroids are irregularly shaped, lacking the gravity to form spherical shapes. Consequently, the light reflecting off an asteroid is unevenly distributed, creating two maxima and two minima in their light curves (Nath, 2024). To determine an asteroid's true magnitude, light curves from multiple observations must be combined. True magnitude provides information about the asteroid's size, with brighter asteroids generally being larger and potentially posing a greater threat upon impact. Additionally, changes in the apparent magnitude of an asteroid can indicate alterations in the asteroid's physical properties and orbital dynamics (Liu et al, 2023). These changes in brightness can be indicative of the success of deflection missions and the potential for future impact events.

Furthermore, the apparent magnitude of an asteroid is a critical parameter for determining its trajectory, potential warning time, and the required deflection strategies in the event of a potential collision with Earth. By accurately measuring the apparent magnitude and monitoring changes in brightness, scientists can assess the potential impact hazard posed by near-Earth asteroids and develop appropriate mitigation strategies (Nath, 2024).

Considering another parameter, the relative velocity of an asteroid is a crucial factor in determining its potential hazard to Earth. When an asteroid approaches Earth with a high relative velocity, the potential impact can have more severe consequences. The kinetic energy of the asteroid increases with higher relative velocity, which can lead to more significant damage upon impact. Additionally, a higher relative velocity can make it more challenging to deflect or mitigate the asteroid's trajectory, increasing the risk of a hazardous collision (Avdyushev et al, 2022). Therefore, the relative velocity of an asteroid is an important consideration in assessing its potential hazard to Earth.

In light of the preceding discussion on the various features of asteroids and the critical parameters influencing their potential hazard to Earth, this research paper seeks to address a pivotal problem: the accurate classification of asteroids as hazardous or non-hazardous. With the looming threat of celestial collisions, accurately assessing the risk posed by asteroids is important for planetary defense.

To tackle this challenge, our study evaluated and compared multiple classification algorithms to determine which achieves the highest accuracy in identifying potentially dangerous asteroids. Through our evaluation process, we enhanced our understanding of asteroid classification and paved the way for further discussion about asteroid mitigation strategies.

## II. Literature Review

### A. *Classifiers*

The goal of a classification algorithm is to correctly identify and label a random input data point within a set of finite categories. Engineering input data is a necessity in generating a robust classification model, and popular approaches involve selecting the most relevant features through correlation analysis or transforming empty values as well as any identified outlier values. One such approach is logistic regression: This classifier models data through a regression model of a selected amount of input features, and transforms that calculated value by using it as input to the sigmoid function:

$$S(x) = \frac{1}{1 + e^{-x}}$$

The output of this sigmoid function is a value between 0 and 1 which essentially represents the "probability" of this datapoint being labeled as one of the two classes(Hosmer 2013). For classification, this value is then compared to some chosen threshold (by default, usually 0.5), to determine the final binary classification of the datapoint. Training model parameters are similar to that of linear regression as all the input feature values have weights that measure the significance of each of the input features of the classifier. These weights can be tweaked through methods including but not subjected to, gradient descent or maximum likelihood estimation.

Similar to the straightforward nature of the Logistic Regression algorithm is the K-nearest neighbors (KNN) algorithm. The algorithm visualizes the input data by mapping it in an n-dimensional space (where n is the number of different attributes of the data). The algorithm utilizes distance functions to categorize data as a certain label based on the proximity between similar data points(Peterson 2009). In terms of nomenclature, KNN illustrates the idea that K number of nearest neighbors are labeled based on the most frequently chosen label within that group of data. Data handling is also crucial in developing an accurate KNN model. Necessary techniques include scaling input data so certain features don't overpower others, as well as filling in empty data points. There are a few common distance functions to choose from when training the algorithm and a prime example of one of

them is calculating the euclidean distance between the two points. To train an effective KNN model, setting the right K value is critical; a K value that is too small has the potential to result in a model that is sensitive to outliers, whereas an exceedingly large K value can lead to oversized groups, causing misclassifications of data.

Unlike the simplicity of the Logistic Regression and KNN algorithms, Decision Tree Classifier (DTC) takes a more structured approach by effectively modeling complex data by ranking each feature's significance. To elaborate on the algorithm's logic: DTC generates a parent node in which each feature is measured based on its corresponding "information gain" which is modeled through the entropy function(Charbuty, Abdulazeez 2021):

$$Entropy(S) = \sum_{i=1}^{c} P_i log 2^{P_i}$$

S is the random feature, c is the number of outcomes for that variable, $P_i$ is the probability of the outcome i

- V(A) is the range of attribute A and and SV is the informational value of attribute V

Once the calculations for each of the previous functions are found, the tree creates a split at that node based on the feature with the highest information gain. The algorithm repeats the previous process until a maximum tree depth is reached based, if specifically selected, or if there is no more information gained from splitting the tree any further. It is crucial to clean input data by filling in empty values and removing irrelevant features to prevent creating an overly complex tree. Techniques to generate a more robust model include: Limiting the amount of splits in the tree, removing–or namely pruning–unnecessary splits in the trees, setting a maximum depth to the tree, and setting a minimum required samples for an output node to be valid and recognized. Following these general guidelines can lead to more accurate results and a better representative model of the given data.

Similar to the DTC, the Random Forest Classifier (RFC) is an extension of its core algorithm. RFC is an ensemble algorithm that generates multiple decision trees with specified input attributes and subsets of the training data(Belgiu, Drăguţ 2016). The algorithm then decides on which tree is the most accurate and chooses that tree as the final implementation to increase the model accuracy and prevent overfitting of the data. Techniques to effectively clean data consist of filling in missing values, scaling or normalizing the input data, as well as encoding categorical attributes into numerical values. The previous techniques are imperative for data preprocessing in regards to the RFC. The algorithm itself entails producing a "forest" of decision trees with a random subset of training data–namely bootstrap sampling–as well as a subset of the number of input features, which is feature bagging. Overfitting is a common issue with decision trees but the concept of ensemble learning by creating a forest of trees and innate randomness by feature bagging and data bootstrap sampling can lead to an accurate and generalizable model.

### *B. Previous Research*

Many of the binary classification methods have been used to examine the problem of hazardous asteroid classification. Starting with Vedant Bahel, he and his group used the NASA Jet Propulsion Small Body dataset to classify an asteroid as potentially hazardous. In order to properly model the data, they examined 4 different classification models: Logistic Regression, K-Nearest Neighbors (KNN), Decision Tree Classifier (DTC), and the Random Forest Classifier (RFC). A 2:1 split was chosen for the training and test data and all hyperparameters were left unset, with the exception of KNN's 'n-neighbor' being set to 3, and the RFC's 'n-estimator' parameter being set to 15. Upon completion of the model's training and testing, the team used 4 metrics to analyze the performance of the model. The first metric was accuracy:

$$accuracy = \frac{|TP| + |TN|}{|observations|}$$

The next two metrics used were precision and recall:

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

$$Recall = \frac{TP}{TP + FN}$$

Finally, they used the F1-score, which is the weighted average of precision and recall. RFC was found to be the most effective model, boasting an accuracy of 99.99%, precision of 99.09%, recall of 99.33%, and F1-score of 99.21%. DTC had very similar results, only slightly falling short of the RFC. Logistic Regression and KNN also reported very high a accuracy, greater than 99%; however, they fell short on precision, recall, and F1-score. They concluded that the Logistic Regression model performed poorly due to the imbalance of the test data, having many more non-hazardous entries than hazardous(Bahel, 2021).

In a similar study, Seyed Matin Malakouti used a dataset from NASA containing 10 columns and 90,000 entries. They chose to use only 7 of the columns: miss distance, relative velocity, absolute magnitude, minimum estimated diameter, and maximum estimated diameter, orbiting body, and sentry object, due to the high impact of each feature. They processed the data further by removing the outliers and scaling the data using a Min-Max scaler. They analyzed the performance of RFC, Extra Tree Classifier, Light GBM, Gradient Boosting, and ADA boost. The data was split into 80% training data and 20% testing data, and the hyperparameters were tuned using grid search. After training, the models were evaluated using the Class of the ROC True, the Class of the ROC False, the micro-average ROC, and the macro-average ROC. Using these evaluation metrics, RFC was found to be the most effective model in classifying both Hazardous and non-hazardous asteroids(Malakouti, 2023).

Kaveti Upender performed a similar analysis on hazardous asteroid classification. They analyzed decision trees and ensemble learning techniques, including RFC, XGBoost Classifier, and Gradient Boosting, on the NASA Small Body dataset, which contain 9 columns and 25,708 rows. During their analysis of the dataset, they chose to drop the column "full name", and to encode the categorical variables to 0 (No) and 1 (Yes). The best model is selected based on the recall, precision, F1-score, and accuracy, and the hyperparameters are trained using grid search. After training and testing concluded, the XGBoost Classifier was found to have higher F1 and accuracy scores than the other models. It reported an F1 Score of 0.98, Recall of 0.97, Precision of 0.98 on Hazardous Asteroids, and an accuracy of 0.9961 (Upender, 2022).

## III. Exploratory Data Analysis

In order to accurately capture the necessary information about asteroids that would pertain to their final classification, the dataset we chose needed to be modified in two primary ways. Upon compiling our .csv file into a pandas DataFrame, there were 2 key steps to ensure that our ultimate dataset was preprocessed and ready to be used in model training: initial DataFrame trimming and feature selection.

### *A. Initial DataFrame Trimming*

The first and foremost step in EDA is getting familiar with the data. The dataset we had selected was initially clean of any unusable or invalid values, and analysis of all data points confirmed no missing values in any of the features as well. This was, in part, one of the reasons why we had selected this dataset: its initial cleanliness and the way it was maintained by the original publisher on Kaggle made it a straightforward task to get the .csv file into a DataFrame and ready to be preprocessed.

With this initial data familiarity task done, we needed to trim the dataset appropriately. This meant observing the columns of the DataFrame and disposing of any that were deemed redundant or moot for the overall classification task at hand. The first step of this was to eliminate any columns which contained non-numerical data. We initially considered potential encoding schemes for these columns, but upon closer inspection we were able to determine that none of the four non-numerical columns in the dataset were essential beyond the preprocessing stage. Each one of these columns is outlined below:

- **Close Approach Date (datatype: string)**
  This column contains string objects representing the date at which the asteroid's close approach with Earth was made. It serves more as an identification mechanism for each asteroid, and thus has no impact on the hazard potential of that given asteroid.
- **Orbiting Body (datatype: string)**
  This column contains string objects representing the body around which the asteroid orbited. Considering that this data was collected by NASA specifically for near-Earth asteroids, naturally this entire column consisted of only a single string: "Earth". Such homogeneity made this column irrelevant to the overall classification task.
- **Orbit Determination Date (datatype: string)**
  Similarly to "Close Approach Date," this column served more as a way to identify each asteroid using the date at which its orbit was determined. This merely sequential data was thus also deemed inconsequential towards the final classification task.
- **Equinox (datatype: string)**
  This column was ultimately removed due to value homogeneity (similarly to "Orbiting Body"). It contained the current standard equinox epoch at the time of recording the asteroid, which is "J2000" for all asteroids. A standard equinox epoch is marked by the Sun (viewed from the Earth) entering one of the 12 constellations on the ecliptic (Durst, 2020). The "J" stands for "Julian epoch", and 2000 marks the first year of this epoch. Again, due to this value being the same for all data points, and also being more of an identification mechanism for each asteroid, this column was removed.

Upon removal of the non-numerical features, the second step of the initial DataFrame trimming was the "domain drops" step. In this step, each remaining feature was considered in the context of existing domain knowledge we had in the field. The driving motives behind each domain drop are outlined below:

- **Neo Reference ID (datatype: int64), Name (datatype: int64), Orbit ID (datatype: int64)**
  These three columns were all simply means of asteroid identification. As described above, such identification contained no useful information towards the task of classifying asteroids.
- **Est Dia in KM (min) (datatype: int64), Est Dia in KM (max) (datatype: int64), Est Dia in Miles (min) (datatype: int64), Est Dia in Miles (max) (datatype: int64), Est Dia in Feet (min) (datatype: int64), Est Dia in Feet (max) (datatype: int64)**
  These 6 columns all contained redundant information about the estimated diameter of each asteroid. The DataFrame contained two additional columns which described the minimum and maximum estimated diameter of the asteroid in meters: Due to unit preference and domain knowledge about average asteroid size and average asteroid size variance, we determined that these six columns could be dropped, whereas the two columns containing this same information in meters were both retained.
- **Relative Velocity in KM per Second (datatype: float64), Relative Velocity in Miles per Hour (datatype: float64)**
  Similarly to the six columns mentioned above, these two columns were redundant unit conversions of a different column, "Relative Velocity in KM per Hour". Again, due to domain knowledge of asteroid velocity, it was determined that the "Relative Velocity in KM per Hour" column contained more appropriate information, and so

its redundant unit-conversion counterparts were dropped.

- **Miss Dist. (Astronomical) (datatype: float64), Miss Dist. (Lunar) (datatype: float64), Miss Dist. (Miles) (datatype: float64)**
  Similarly to many of the aforementioned columns, these columns were mere unit-conversions of a more informative column, "Miss Dist (KM)." They contained no additional information, and based on domain knowledge of near-Earth distance units, as well as unit preference and consistency, we resolved to retain "Miss Dist (KM)" and dispose of its unit-conversion counterparts.

In total, this initial trimming enabled a 45% reduction in the amount of dataset features, from an initial 40 features to a final 22 features. This reduction of feature clutter was crucial for the next step of the EDA process: model feature selection.

### *B. Feature Selection*

After cleaning up the data, we understood quickly that there existed a certain subset of features remaining which had the potential for high predictive power the final value of the target label for each asteroid. In order to obtain these features, we iterated over the absolute values of the correlations of all features with the target label, and selected the top six features to be used for our final model. The reasoning behind choosing six features was not arbitrary: beyond these top six, the correlation values for the remaining features were negligible. Domain knowledge also reinforced our confidence in our selected features, as many of the dropped features did not appear to be very related to an asteroid's hypothetical hazard potential. The six final features selected for our model are listed below:

- Absolute Magnitude
- Eccentricity
- Minimum Orbit Intersection
- Orbit Uncertainty
- Perihelion Distance
- Relative Velocity in km per hr

To assert the chosen features' relevance to the problem at hand, we utilized the Seaborn library to visualize how each of the features related to the target label. These results are summarized in figure 1 below.

### *C. Random State Computation for train_test_split()*

Upon selecting our features, the next step in the preprocessing stage was to split the remaining features into training and testing sets. This process was initially very straightforward: we simply used the train_test_split() function from the scikit-learn library with a test size of 20% and a pseudo-randomized shuffling of the data points using the function's "shuffle" parameter.

In order to get the data points shuffled, we also needed to select a value for the function's random_state parameter. Initially, we chose a random number (17) and proceeded with the training procedure. Upon further scrutiny of the EDA code, however, it became apparent that this random_state parameter was much more impactful on the final model results than what met the eye. By shuffling the data points according to a different pseudo-random scheme with each random_state value, we were in effect changing the model's training data and thus leading to fluctuations in our resulting accuracy values.

In order to account for this, we decided to come up with a scheme to find the ideal random_state parameter value. We understood that the ratio of hazardous to non-hazardous asteroid data points in the train and test dataset needed to closely match the same ratio in the original dataset, so as to offer the model the same data trends represented in the overall dataset. To do this, we simply looped over random_state values from 0 to 999, calling train_test_split() on the dataset at each iteration and observing the ratio between hazardous and non hazardous data points. This ratio was then compared to the ratio observed in the DataFrame. The loop kept track of which random_state value offered the lowest difference in ratios: This value turned out to be random_state = 8, and the loop results are outlined below. After random_state = 8, no ratio was able to match the original DataFrame ratio more closely, meaning after this value the loop did not continue to update the best random state value. **Table 1** below depicts the output of this loop, with the ratio of the DataFrame at the top of the image, and the lowest-ratio-difference random_state parameter values from within the loop below. Finally, the best random_state value found was printed by the loop at termination.

| Random State | Ratio Difference |
|---|---|
| 0 | 0.099164358167673914 |
| 1 | 0.00910086568919643 |
| 2 | 0.00441432931599913 |
| 3 | 0.0031676209787437086 |
| 6 | 0.001659121667329454 |
| 8 | 0.0001463895114304133 |

**Table 1**: Ideal random state values based on ratio difference.

At termination of this procedure, we were confident that using the pseudo-random split granted by random_state=8 would ensure optimal model results. It appeared, however, that despite the rationale of this random_state value and the procedure which birthed it, this particular data split did not yield accuracies on par with some of the best results reached by our final models.

This process was then completely nullified by the re-introduction of the initial random_state parameter, random_state=17. With random_state=17, the data split was determined to be ideal for the model's final performance, as it was able to reach higher accuracies under this split. Thus, the procedure for selecting the best random_state parameter was a means for us to attempt to rationalize the final value of random_state which ultimately (and perhaps unironically) fell short of the truly ideal random_state value: The random number 17.

The final step in data preprocessing was then simply scaling the data in preparation for training. At this point, our data was cleaned, split into training and testing, standardized, and ready for actual processing. The crucial EDA phases granted

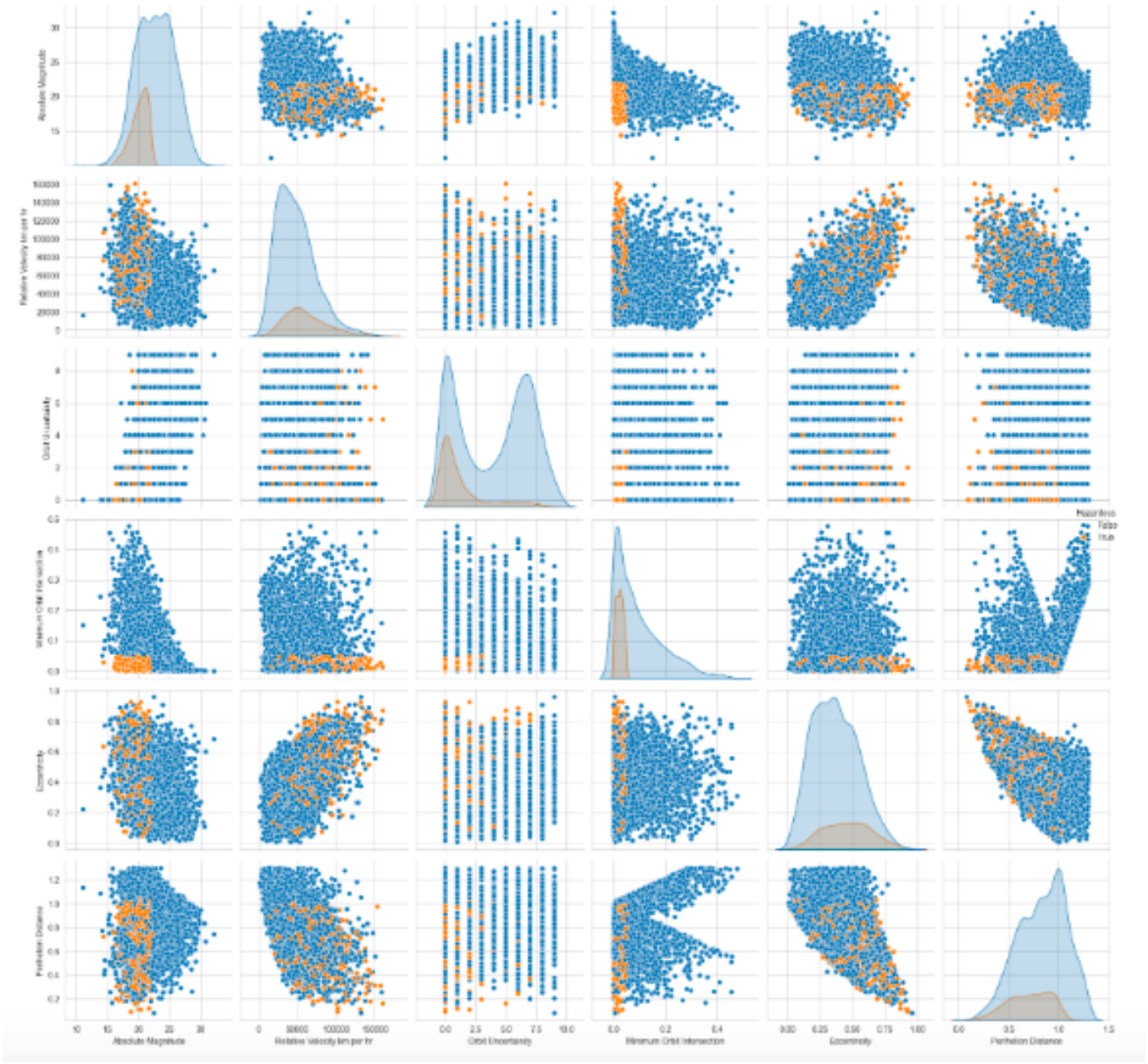

Fig. 1: Visualizations of the top six features. In blue are the non-hazardous asteroids, and in orange are the hazardous ones.

us an enhanced perspective on the data overall, giving us the confidence and understanding required to use this data for model training.

## IV. Proposed Methodologies

To classify asteroids as hazardous or non-hazardous, we propose a performance comparison of a variety of model architectures. Using classic machine learning model architectures from the scikit learn library, we sought to compare how various classification models would perform with this asteroids dataset. The objective was simple: keep the best model.

We selected four well-known classification model architectures to compare: Logistic Regression, K-Nearest-Neighbors (KNN), Decision Tree Classifier (DTC), and Random Forest Classifier (RFC). Each of these model architectures was trained on the same train dataset curated during our preprocessing stage.

For the Logistic Regression classifier, seeing as how it takes no hyperparameters, training always resulted in identical accuracies on the train/test datasets. We then focused our attention on finding the best hyperparameters for the remaining three model architectures, which we did using grid search. Looping over a predetermined range of 2 through 10 and

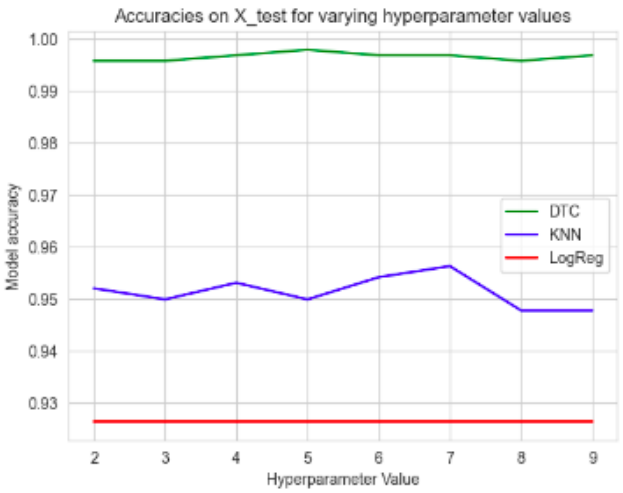


Fig. 2: Test accuracy for DTC, KNN, and Logistic Regression across hyperparameter range of [2, 10]

using the loop variable as the hyperparameter value, we trained KNN and DTC and plotted their accuracies given the changing parameters. For KNN, the hyperparameter served as k (the number of neighbors), while for DTC it served as the maximum depth the tree was allowed to reach.

The training procedure for RFC necessitated a different approach, seeing as how the RFC algorithm has two hyperparameters instead of one (max_depth, or the max depth each tree is allowed to reach, and n_estimators, the number of trees the model will have). To find the best pair of parameters, we utilized RandomizedSearchCV(), a function which randomly iterates over a predetermined parameter space and picks the parameters found to yield the highest model accuracy. We defined the max_depth range as [1, 20], and the n_estimators range as [50, 500]. We then used the resulting hyperparameters of the RandomizedSearchCV() as the middle points for new max_depth and n_estimators ranges: These ranges were then fed to a traditional grid search algorithm.

For each model architecture, the hyperparameters which led to the model with the highest accuracy were saved, and the final accuracies of all four best models (one of each architecture) were then compared.

## V. Experimental Results and Evaluations

The first three models, Logistic Regression, KNN, and DTC, had their accuracies directly compared over the aforementioned range of hyperparameter values used within the grid search. During the grid search, the models were each tested on the scaled test data. Their accuracies throughout the grid search, as well as each model's highest accuracy reached, are summarized below:

The 99.79% accuracy reached by the DTC stood out as the best accuracy reached by these three model architectures. This model's accuracy on the test data is described in further detail by the classification report and confusion matrix for this model, summarized in table 3 and figure 3 below.

| Model | Accuracy |
|---|---|
| KNN (k = 7) | 0.9562899 |
| DTC (depth = 5) | 0.9978678 |
| Logistic Regression | 0.9264392 |

**Table 2**: KNN, DTC, LR: best accuracy scores.

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| **False** | 1.00 | 1.00 | 1.00 | 793 |
| **True** | 1.00 | 0.99 | 0.99 | 145 |
| **accuracy** | N/A | N/A | 1.00 | 938 |
| **macro avg** | 1.00 | 0.99 | 1.00 | 938 |
| **weighted avg** | 1.00 | 1.00 | 1.00 | 938 |

**Table 3**: Classification report for the DTC model. Accuracy score = 0.9978678.

For the RFC model, after the RandomizedSearchCV() function the best values for max_depth and n_estimators were 7 and 322, respectively. Using those parameters in a traditional grid search, we obtained improved hyperparameters of 12 and 323, respectively. These final hyperparameters were then used to train the RFC model, which was able to reach a final accuracy of roughly 99.89%. This is summarized in the model's classification report and visualized in the model's confusion matrix in table 4 and figures 4 below, respectively.

Keeping in mind the limited dataset with which we were working (approximately 4700 data points in the entire dataset), both our RFC and DTC architectures were able to surpass the results reached by contemporary published research on the topic. When compared to the more complex XGBoost architecture utilized by Upender et al. in 2022, our RFC and DTC were both able to reach superior accuracies: As discussed in our literature review, Upender et al. reached an accuracy of approximately 99.69% using 25,700 data points, lower than our obtained accuracies of 99.79% and 99.89% using our DTC and RFC architectures, respectively. A paper with a similar goal published by Seyed Matin Malakouti, Mohammad Bagher Menhaj, and Amir Abolfazl Suratgar on Elsevier in 2023 was only able to reach consistent test accuracies of 91% using an RFC architecture: we surpassed this not only with our RFC and DTC models, by even using our KNN and logistic regression models, which were each able to reach 95.6% and 92.6% accuracies, respectively.

It was of particular interest for us to rationalize why the RFC and DTC models performed so much better than the KNN and logistic regression models, as well as why the RFC model was able to edge out the DTC model in accuracy after training completion. We propose several hypotheses as to why the models ranked as they did in terms of accuracy and overall performance:

- Logistic Regression is, when compared to the other classification algorithms, relatively simplistic. Ideally, this model architecture would be used on a dataset which had logistic patterns within it. It still is capable of reaching decent classification accuracy, but it does not hold up

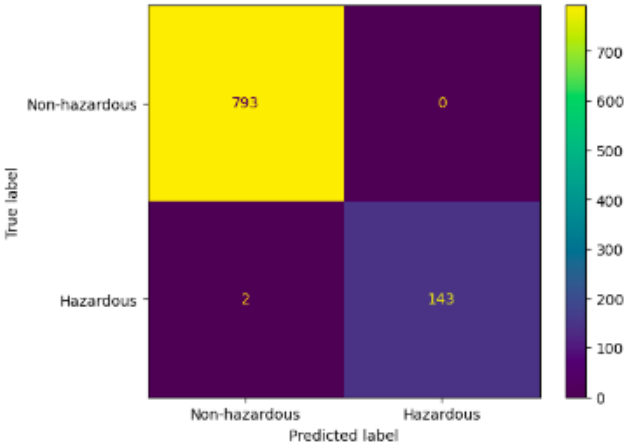


Fig. 3: Confusion matrix for the DTC model

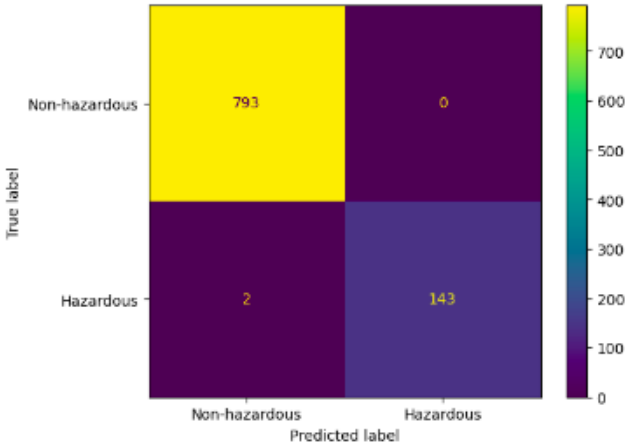


Fig. 4: Confusion matrix for the RFC model

to the three other model architectures likely due to its intrinsic simplicity.

- KNN could potentially be suffering from the curse of dimensionality: Attempting to compare euclidean distances of points on a 6-dimensional plane could be a contributing factor to this architecture ultimately not performing as well as DTC and RFC. DTC and RFC suffer less from this factor due to the intrinsic nature of their decision making: The binary decision splitting would be less affected by increasing dimensions than KNN with its distance computations.
- By observing fig. 1, it also becomes imminent that the hazardous/non-hazardous data points are highly interspersed across many of the features. This could also be detrimental to KNN's final performance, as decision boundaries may become more difficult to draw given the relative spatial position of each datapoint coordinate.

| | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| **False** | 1.00 | 1.00 | 1.00 | 793 |
| **True** | 1.00 | 0.99 | 1.00 | 145 |
| **accuracy** | N/A | N/A | 1.00 | 938 |
| **macro avg** | 1.00 | 1.00 | 1.00 | 938 |
| **weighted avg** | 1.00 | 1.00 | 1.00 | 938 |

**Table 4**: Classification report for the RFC model. Accuracy score = 0.9989339.

- RFC slightly edges out DTC likely due to the increased decision-making scale introduced by the RFC algorithm. The DTC is clearly reaching very high accuracies, but RFC takes advantage of multiple DTCs, aggregating their results and thus reaching better accuracies, especially on a more complex dataset. It is of important note that this entails the RFC model would have greater computational demands than the DTC: Given a situation in which computational complexity took precedence over minute accuracy discrepancies (such as, for example, aboard potential interstellar vessels attempting to categorize asteroids in real time), it may be beneficial to use our DTC architecture in place of our RFC.

## VI. Conclusion and Discussion

The two main objectives of this project were to determine how to effectively model whether or not an asteroid posed a threat to Earth. Once we generated accurate models we wanted to establish which model, out of the selected models we chose to train, was the most fit at modeling the data. Among all the trained classifiers, RFC was the most accurate model in determining the proper label for a given asteroid, with a model accuracy score of 99.89%. However, it is important to note that our DTC model, with an accuracy of 99.79%, performs better than currently published research for DTCs, including the works of Upender and Malakouti. Despite the model's accuracy, we encountered some limitations to its applications. These limitations are elucidated by the fact that the model works retroactively, meaning that making an accurate prediction requires the asteroid to be in close proximity to Earth. This also implies that we are unable to make effective predictions on asteroids that are several light years away. However, there exists some plausible solutions. For instance, we can extend our model to the field of computer vision directly, training future models to capture images of approaching asteroids. By capturing these images light years in advance, we can dynamically train our model to serve its intended purpose, to educate the public of the imminent dangers that lie within the skies above.

## VII. Github Link

https://github.com/nsaadat01/ecs-171-project

## VIII. Project Roadmap: ECS171 Final Project

### A. Vision and Goals

- **Vision**: Accomplish a project that solves a real-world issue while we all learn about different machine learning techniques.
- **Goals**:
  - Learn about exploratory data analysis (EDA).
  - Learn about multiple machine learning classifiers.
  - Implement version control practices.
  - Develop skills in web development.

### B. Milestones

1) **Project Kickoff** (April 12, 2024)
2) **Find a dataset** (April 16, 2024)
3) **Decide on dataset** (April 20, 2024)
4) **Set up GitHub repository** (April 20, 2024)
5) **Clean up data, select features** (April 30, 2024)
6) **Train different models, tune hyperparameters, export best model** (May 7, 2024)
7) **Start Building Website** (May 7, 2024)
8) **Start Writing the Paper** (May 7, 2024)
9) **Launch Website** (May 31, 2024)
10) **Record demo video** (June 7, 2024)
11) **Finish the paper** (June 8, 2024)

### C. Deliverables

- Complete GitHub repository
- Fully functional website
- Demo video for website
- Paper exploring our process

### D. Timeline

| Phase | Start Date | End Date |
|---|---|---|
| Project Kickoff | April 12, 2024 | April 12, 2024 |
| Find a Dataset | April 12, 2024 | April 16, 2024 |
| Decide on Dataset | April 17, 2024 | April 20, 2024 |
| Set up GitHub Repository | April 20, 2024 | April 20, 2024 |
| Clean up Data and Select Features | April 21, 2024 | April 30, 2024 |
| Train Models, Tune Hyperparameters | May 1, 2024 | May 7, 2024 |
| Start Building Website | May 7, 2024 | May 31, 2024 |
| Start Writing the Paper | May 7, 2024 | June 8, 2024 |
| Launch Website | May 31, 2024 | May 31, 2024 |
| Record Demo Video | June 1, 2024 | June 7, 2024 |

### E. Tasks and Activities

- **Planning and Research**: Define project scope, assign roles, and gather initial resources.
- **Data Collection and EDA**: Collect relevant data, perform exploratory analysis, visualize data, and summarize findings.
- **Model Selection and Implementation**: Research and choose appropriate machine learning classifiers, implement models, and evaluate performance.
- **Version Control**: Set up a version control system (Git), establish collaboration protocols, and ensure regular commits.
- **Web Development**: Design and develop a web application to showcase the project, integrate machine learning models, and ensure mobile responsiveness.
- **Testing and Refinement**: Conduct thorough testing, fix bugs, optimize performance, and refine the web application and models.

### F. Resources

- **Time**: 3 months
- **Tools**:
  - Development tools: Visual Studio Code, Colab
  - Version Control: Git
  - Data Analysis and Machine Learning: Python, Pandas, SciKit-Learn
  - Web Development: HTML, CSS, JavaScript, Flask

### G. Dependencies

- **Data Cleaning and Feature Selection**: Can only begin after the dataset is chosen.
- **Training Models and Tuning Hyperparameters**: Dependent on the completion of data cleaning and feature selection.
- **Building the Website**: Can only begin after the final model is fully trained and exported.
- **Writing the Paper**: Can start concurrently with model training but will depend on having initial results from the exploratory data analysis.
- **Launching Website**: Dependent on the completion of website development and initial testing.
- **Recording Demo Video**: Dependent on the website being fully functional and launched.
- **Finalizing the Paper**: Dependent on the completion of model training, website development, and initial analysis results.

### H. Risks and Challenges

- **Scheduling conflicts**: Potential conflicts with busy schedules.
- **Technical Issues**: Possible unforeseen technical challenges during development.
- **Learning Version Control**: Ensuring no data is lost or mistreated during version control.

### I. Progress Tracking

- **Weekly Status Meetings**: Regular updates and check-ins with the project team.
- **Milestone Review**s: Assess progress at each milestone.
- **Project Management Software**: Use of tools like WhatsApp and GitHub to set up meeting times and track progress.

## References

[1] Avdyushev, V., Syusina, O., & Tamarov, V. (2022). Stochastic simulation of orbital uncertainty of potentially hazardous asteroids observed in one appearance. Planetary and Space Science, 216, 105488. doi:10.1016/j.pss.2022.105488

[2] Bahel, V., Bhongade, P., Sharma, J., Shukla, S., & Gaikwad, M. (2021). Supervised Classification for Analysis and Detection of Potentially Hazardous Asteroid. 2021 International Conference on Computational Intelligence and Computing Applications (ICCICA), Nagpur, India, 2021, pp. 1-4, doi: 10.1109/ICCICA52458.2021.9697222.

[3] Belgiu, Mariana, and Lucian Drăguţ. "Random Forest in Remote Sensing: A Review of Applications and Future Directions." ISPRS Journal of Photogrammetry and Remote Sensing, vol. 114, no. 114, Apr. 2016, pp. 24–31, www.sciencedirect.com/science/article/pii/S0924271616000265, https://doi.org/10.1016/j.isprsjprs.2016.01.011.

[4] Charbuty, B., & Abdulazeez, A. (2021). Classification Based on Decision Tree Algorithm for Machine Learning. Journal of Applied Science and Technology Trends, 2(01), 20 - 28. https://doi.org/10.38094/jastt20165

[5] Durst, S. (2020). Aquarius Equinox Epoch and Precession History. NASA ADS, 235, 118.05. https://ui.adsabs.harvard.edu/abs/2020AAS...23511805D/abstract#:~ text=Equinox%20epochs%20are%20astronomical%20periods

[6] Hosmer, David W. Jr, et al. Applied Logistic Regression. Google Books, John Wiley & Sons, 26 Feb. 2013, books.google.com/books?hl=en &lr= &id=bRoxQBIZRd4C&oi=fnd&pg=PR13&dq=logistic+regression& ots=kM1Rvu4Xd7&sig=D5CvcHHdLpYLVl_uZaNGC87KAcw#v =onepage&q= logistic%20regression&f=false. Accessed 5 June 2024.

[7] Jin, Z. H. U., Zhi-tao, Y., Hai-bin, Z., Zhuo-xi, H. U. O., & Xiao-jun, J. (2022). Short-term Hazardous Asteroid. Chinese Astronomy and Astrophysics, 46(2), 55–64. doi:10.1016/j.chinastron.2022.05.005

[8] Liu, S., Wu, Z., Yan, J., Gao, J., Huang, H., Cao, J., … Barriot, J.-P. (2023). Orbital eccentricity and inclination distribution of main-belt asteroids—The Statistical model revisited. Icarus, 404, 115650. doi:10.1016/j.icarus.2023.115650

[9] Malakouti, S. M., Menhaj, M. B., & Suratgar, A. A. (2023). Machine learning techniques for classifying dangerous asteroids. MethodsX, 11, 102337. https://doi.org/10.1016/j.mex.2023.102337

[10] Nath, A. (2024). Developing algorithms to determine an Asteroid's physical properties and the success of deflection missions. Acta Astronautica, 220, 62–74. doi:10.1016/j.actaastro.2024.04.019

[11] Peterson, Leif. "K-Nearest Neighbor." Scholarpedia, vol. 4, no. 2, 2009, p. 1883, www.scholarpedia.org/article/K-nearest_neighbor, https://doi.org/10.4249/scholarpedia.1883.

[12] Salotti, J.-M. (2022). Humanity extinction by asteroid impact. Futures, 138, 102933. doi:10.1016/j.futures.2022.102933

[13] Upender, K., Krishna, T. S., Pothanna, N., & Kumar, P. V. S. (2022). Predicting the Potentially Hazardous Asteroid to Earth Using Machine Learning. Proceedings of Second International Conference on Advances in Computer Engineering and Communication Systems, 359–369. https://doi.org/10.1007/978-981-16-7389-4_34