ASTEROID THREAT DETECTION USING MACHINE LEARNING

Noor Hadiya

Department of Computer Science

Engineering (Data Science)

[noorhadiya23@gmail.com](mailto:noorhadiya23@gmail.com)

Chaithra Shree P

Department of Computer Science

Engineering (Data Science)

[chaithraprakashdm@gmail.com](mailto:chaithraprakashdm@gmail.com)

Abstract:

The identification of asteroidal threats is a vital component of Earth's planetary defense strategy. The accuracy and effectiveness of traditional approaches for predicting asteroid trajectories are limited since they rely on astronomical observations and computations. The use of data-driven strategies to improve detection, classification, and prediction capacities is the main emphasis of this report's investigation into the application of machine learning (ML) techniques in asteroid danger detection. The goal of this work is to increase the precision and dependability of asteroid threat detection systems through the analysis of many datasets and the use of machine learning methods.

Introduction:

Background on Asteroid Impacts and Their Potential Consequences:

One of the biggest natural threats to life on Earth is the impact of asteroids. Asteroids of all sizes have struck Earth throughout history, with disastrous results including mass extinctions, tsunamis, and changes to the climate. The collision of the Chicxulub asteroid, which is thought to have led to the extinction of the dinosaurs some 66 million years ago, is one of the most significant occurrences in recent history. Even though huge asteroid impacts are uncommon, smaller impacts can nonetheless endanger human life and seriously damage infrastructure.

The possible repercussions of asteroid impacts highlight how crucial it is to create effective detection and mitigation plans. For the purpose of implementing mitigation measures and providing enough warning time to prevent any impacts, early detection of potentially harmful asteroids is imperative.

Overview of Traditional Asteroid Detection Methods:

Astronomical observations made using space-based observatories and ground-based telescopes have

historically been the basis for asteroid detection. By monitoring the positions of celestial objects, such as asteroids, with respect to the background stars over time, astronomers are able to trace their movement. Astronomers can determine the trajectories of asteroids and forecast their future routes by examining these data.

Important conventional techniques for asteroid detection consist of:  
  
1. Telescopic Surveys: Systematic, large-scale surveys using telescopes to look for asteroids.

2.  Astrophysical Models: Mathematical models that forecast asteroidal orbits based on observable data.

3.  Impact Monitoring: Systems for tracking potentially dangerous asteroids and determining the degree of threat they pose.

Furthermore, the increasing quantity of near-Earth objects (NEOs) pose’s difficulties for conventional detection techniques, emphasizing the necessity for creative ways to improve detection performance.

Introduction to the Application of Machine Learning in Asteroid Threat Detection:

A powerful tool for increasing the precision and effectiveness of asteroid danger warning systems is machine learning (ML). ML algorithms don't require explicit programming in order to evaluate massive amounts of data, spot trends, and provide predictions. When it comes to asteroid danger identification, machine learning algorithms have various benefits, such as:

1. Data-driven Approach: To improve detection skills, machine learning algorithms can make use of a variety of datasets, such as observational data, orbital parameters, and physical features of asteroids.
2. Pattern Recognition: By recognizing intricate patterns and irregularities in asteroid data, machine learning algorithms make it possible to detect hazards that were previously undetected.
3. Automation: ML models can automate a number of detection-related operations, lightening the astronomers' workload and increasing productivity.
4. Adaptability: ML algorithms are able to change and grow with new data, which enables detection systems to get better over time.

Researchers hope to improve our capacity to identify, categorize, and forecast asteroid trajectories by utilizing machine learning (ML). This will help to advance the creation of more potent planetary defense tactics. This research investigates the use of machine learning techniques for asteroid danger detection and looks at how they might advance our knowledge and help mitigate asteroid threats.

Dataset:

NASA - Nearest Earth Objects

A cumulative data for Nearest Earth Objects by NASA

There is an infinite number of objects in the outer space. Some of them are closer than we think. Even though we might think that a distance of 70,000 Km cannot potentially harm us, but at an astronomical scale, this is a very small distance and can disrupt many natural phenomena. These objects/asteroids can thus prove to be harmful. Hence, it is wise to know what is surrounding us and what can harm us amongst those. Thus, this dataset compiles the list of NASA certified asteroids that are classified as the nearest earth object.

This file contains various parameters/features based on which a particular asteroid that is already classified as nearest earth object may or may not be hazardous.

It consists of comets and asteroids whose orbits approach those of Earth. Due to their possible impact hazard and scientific worth, these objects are of interest to astronomers and scientists.  
  
Known as "Spaceguard," the NASA Near-Earth Object Program keeps track of and catalogs these objects. Finding, monitoring, and characterizing NEOs—especially those that might be dangerous to Earth—is the main objective of the program. NASA tracks NEOs and evaluates their orbits using a number of ground-based observatories and telescopes located all over the world.

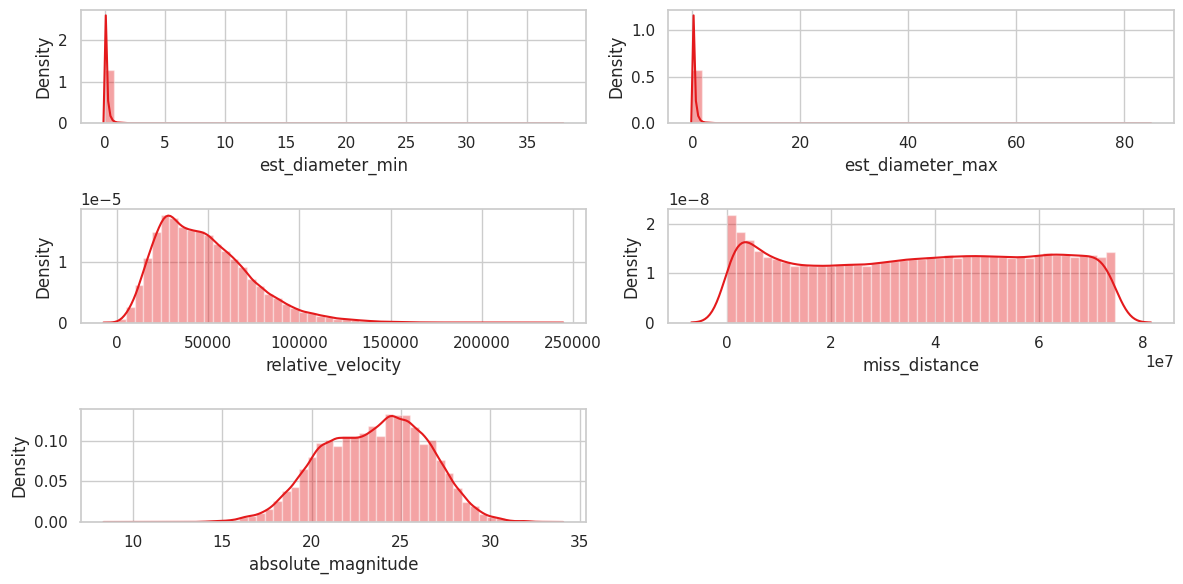
Details on the object, including its size, mass, composition (if known), and likelihood of an impact, are available in the NASA NEO database. In order to evaluate the risk that NEOs pose and to design possible mitigation techniques in the event that a hazardous object is discovered, this data is essential.

Data Preprocessing Techniques:

The asteroid data must be cleaned and prepared for analysis through preprocessing procedures before machine learning methods can be applied to it. Typical methods of preprocessing consist of:

1. Data Cleaning: Erroneous data points, such as outliers, missing values, and inaccuracies brought on by observational mistakes, are eliminated or corrected.
2. Normalization: To guarantee that every variable contributes equally to the analysis, scale the dataset's features to a standard range. Larger scale features are kept out of the model training process by normalization.
3. Feature Extraction: Extracting useful features that are instructive for asteroid detection from the raw data. In order to build new representations that effectively capture significant patterns in the data, feature extraction approaches may include combining or altering preexisting features.
4. Dimensionality Reduction: Keeping as much pertinent information as feasible while reducing the number of characteristics in the dataset. To lower the dimensionality of high-dimensional datasets, methods like Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) can be applied.
5. Imputation: Using strategies like mean imputation, median imputation, or more sophisticated approaches like k-nearest neighbors (KNN) imputation, one can fill in the missing values in the dataset.

We try to analyse our dataset by applying some of these techniques and get results and the methods are:



DISTRIBUTION PLOTS (DIST-PLOT):

Dist-plots are an important exploratory data analysis tool that help with feature selection, preprocessing, understanding of the data, and model interpretation in ML-based asteroid detection projects. They aid in the development of ML models for the identification of asteroids and other astronomical objects that are more reliable and accurate.

Deductions to be made are:

* Follows Gaussian distribution
* so, standard scaler is better
* High skewness in est\_diameter\_min and est\_diameter\_max

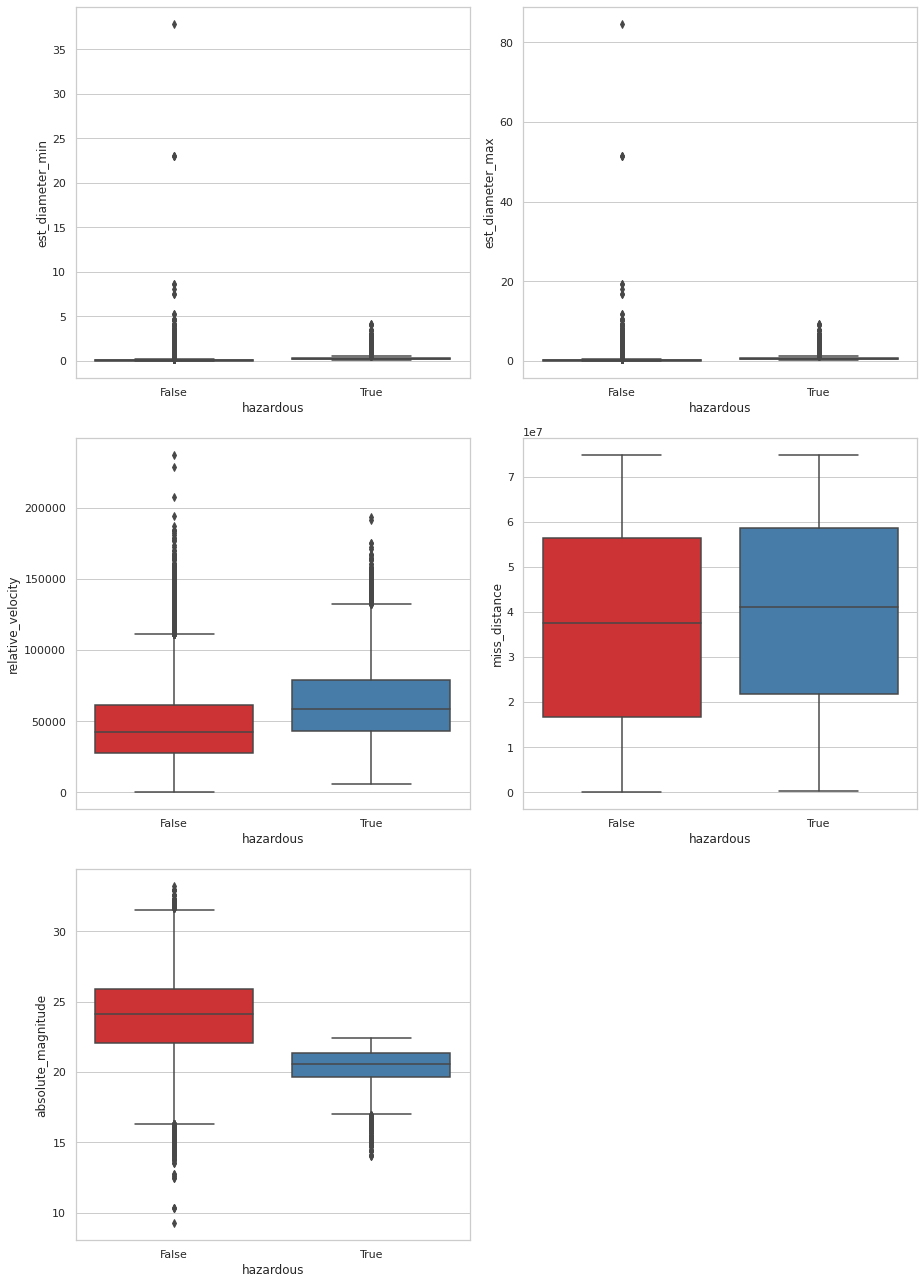
BOX PLOT:

In ML-based asteroid detection projects, box plots enhance other visualization methods by offering insights into data distribution, spotting outliers, evaluating feature relevance, and supporting model interpretation and validation. They are essential to comprehending the dataset and developing trustworthy machine learning models for asteroid identification and categorization.

From this box plot we can:

* Identify outliers
* Compare feature distributions
* Check assumptions
* Accessing feature importance
* Validation and quality assurance
* Model interpretation

Deductions to be made are:

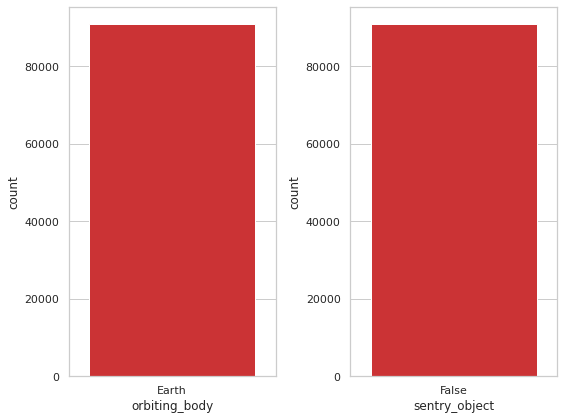
* Need to remove outliers

COUNT PLOT:

Count plots are a useful tool for data exploration, feature analysis, model evaluation, communication of results, and display of class distribution in machine learning asteroid detection projects. They aid in the development of ML models for asteroid detection and classification that are precise and easy to understand.

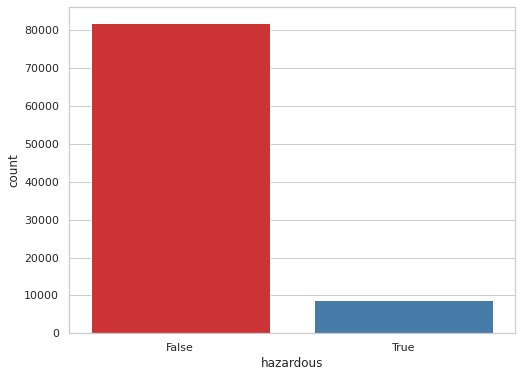
Count plots can be used for:

* Class distribution visualization
* Feature vs class visualization
* Data understanding and exploration
* Feature importance assessment
* Communication of results



Reductions:

* Only 1 unique value in these columns
* No impact on the label
* Can be dropped

DATA IMBALANCE:

This bar graph gives us the count of how true the value is and how false the value is of an asteroid being hazardous.

This is done by plotting it to count versus the hazardous value.

HEAT MAP:

Heat maps are used in asteroid detection for:

Correlation Analysis: Heat maps work well for showing how various features (variables) in the dataset are correlated. Knowing the correlations between different features in asteroid detection projects can help determine which features are most important for forecasting the qualities of asteroids or differentiating between different types of asteroids.

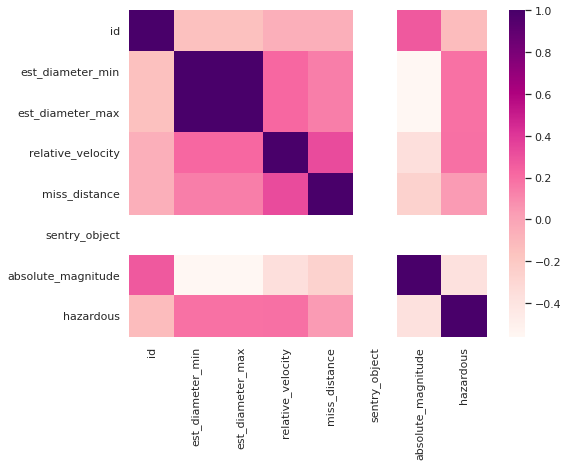
Feature selection: Heat maps can be used to select features by pointing out features that are highly linked or redundant. Eliminating these characteristics can result in more resilient machine learning models by lowering multicollinearity and complexity.

Data preprocessing: Using heat maps, you can find outliers or missing values in the dataset. To guarantee the accuracy and dependability of the data used to train machine learning models, data

preprocessing techniques like imputation and outlier removal can be informed by visualizing the distribution of missing values or outliers across several characteristics.

Analysis of Model Performance: By displaying confusion matrices or prediction errors, heat maps can be used to examine how well machine learning models are performing. For instance, in classification tasks, a heat map of the confusion matrix can help identify classes that are misclassified a lot and identify regions where the model needs to be strengthened.

Parameter Tuning: By illustrating how models perform at various parameter values, heat maps can help with hyperparameter tuning for machine learning algorithms. To help data scientists choose the best parameter values for their models, heat maps, for example, can illustrate how changes in model parameters effect metrics like accuracy, precision, recall, or F1-score.



In general, heat maps are an effective visual assistance for correlation analysis, feature selection, data preparation, model performance analysis, parameter adjustment, and model interpretation in machine learning projects involving asteroid detection. They aid in the development of ML models for asteroid detection and categorization that are more precise and understandable.

DATA PREPROCESSING:

Encode categorical features:

The act of transforming categorical data into a numerical representation that machine learning algorithms may use is known as "encoding categorical features." Qualitative variables with distinct categories or levels, like "color" (red, blue, green) or "type" (cat, dog, bird), are represented by categorical data.

This can be used for:

* Numerical representation
* Avoiding misinterpretation
* Maintaining consistency
* Improving model performance
* Handling algorithms

Since most machine learning algorithms require numerical input data, categorical features must be encoded. There are numerous widely used methods for encoding categorical features, including:

One-Hot Encoding: Using this technique, each category in the original feature has a binary column created with a "1" denoting the category's presence and a "0" denoting its absence. It works well when there is no intrinsic rank or order among the categories.   
  
Label Encoding: By giving each category a distinct integer, this method effectively turns them into ordinal variables. When a categorical characteristic has an innate order or rank, it makes sense.   
  
Ordinal Encoding: This method of assigning integers to categories maintains the order of the categories, much like

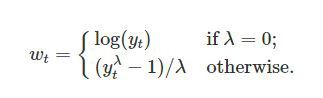
label encoding. For ordinal categorical variables, it is appropriate.

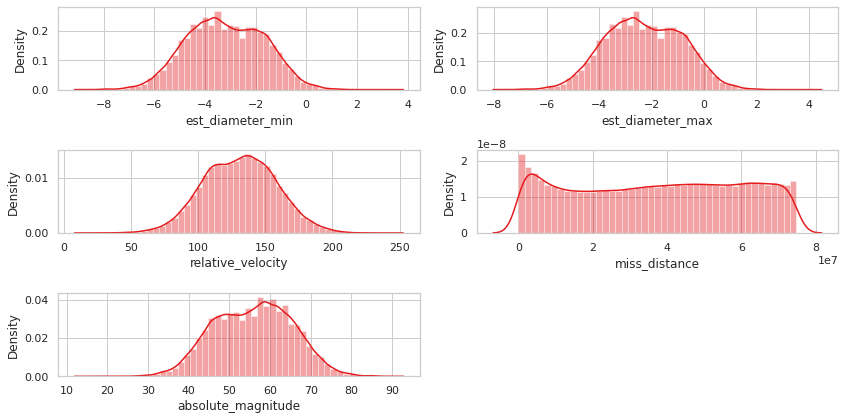
Here we use label encoding for our model.

DATA NORMALIZATION:

Box-Cox Transformation:

The Box-Cox Transformation is a powerful tool in data pretreatment and analysis because it may be used to normalize data, stabilize variance, improve model performance, enhance interpretability, and lessen the impact of outliers. It's important to remember that the Box-Cox Transformation makes the assumption that the data is positive and devoid of zeros. Alternative transformations like the Yeo-Johnson Transformation can be more suited if the data contains zeros or negative numbers.



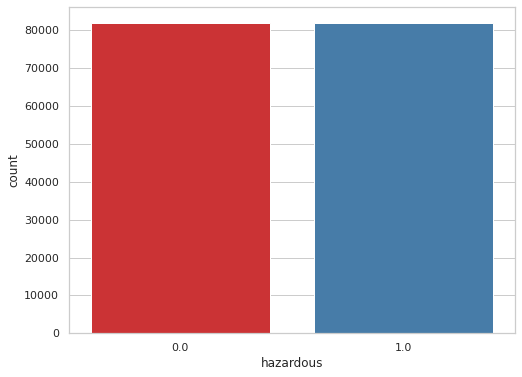


After applying this we plot a dist-plot to get the normalized results.

Null-Value Imputation

Null value imputation, or missing value imputation, is a technique used to handle missing data in a dataset. Missing data is a common issue in real-world datasets and can arise due to various reasons such as data collection errors, equipment malfunction, or simply because certain information was not collected.

Null value imputation is used to replace missing values with estimated or predicted values in order to make the dataset complete and suitable for analysis.

Here we apply “KNNImputer” which is a method used for imputing missing values in a dataset using the k-nearest neighbors algorithm. It is available in the scikit-learn library.

Handling Imbalanced Data: Upsampling

Upsampling, also known as oversampling, is a technique used in machine learning to address class imbalance in a dataset. Class imbalance occurs when one class (the minority class) is significantly underrepresented compared to another class (the majority class). This can lead to biased models that perform poorly in predicting the minority class.

Upsampling involves increasing the number of instances in the minority class to balance the class distribution.

Methods for upsampling:

* Random oversampling
* SMOTE (Synthetic Minority Over-sampling Technique)
* ADASYN (Adaptive Synthetic Sampling)
* Borderline-SMOTE

Here we apply SMOTE method for our model.

SMOTE generates synthetic instances for the minority class by interpolating between existing instances. It creates new instances by selecting a random set of similar instances from the minority class and then randomly combining features to create synthetic samples along the line segments joining any/all of the k minority class nearest neighbors.

When we apply a count plot we get the graph

* SMOTE synthesises new minority instances
* generates the virtual training records by linear interpolation
* generated by randomly selecting one or more of the k-nearest neighbors

**SCALE DATASET:**

**Standard Scaler:**

Where,

**APPLYING ML MODELS:**

1. **AdaBoost classifier:**

**Before resampling when we train our model we get the accuracy and precision scores as:**

Accuracy of predictions on Train set: 90.95 %

Accuracy of predictions on Test set: 90.94 %

Precision of predictions on Train set: 71.83 %

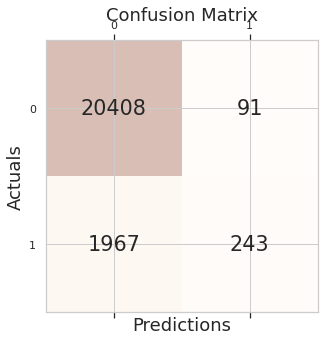
Precision of predictions on Test set: 72.75 %

Recall of predictions on Train set: 11.46 %

Recall of predictions on Test set: 11.0 %

f1 score of predictions on Train set: 19.77 %

f1 score of predictions on Test set: 19.1 %

The confusion matrix we get is:

**After resampling when we train our model we get the accuracy and precision scores as**:

Accuracy of predictions on Train set: 86.53 %

Accuracy of predictions on Test set: 86.55 %

Precision of predictions on Train set: 81.53 %

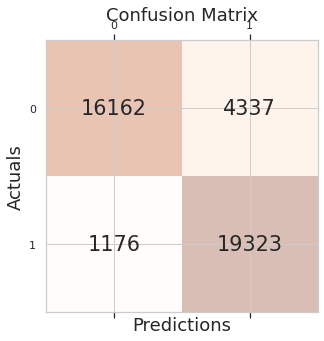
Precision of predictions on Test set: 81.67 %

Recall of predictions on Train set: 94.44 %

Recall of predictions on Test set: 94.26 %

f1 score of predictions on Train set: 87.51 %

f1 score of predictions on Test set: 87.52 %

Confusion matrix:

1. Random forest:

**Before resampling when we train our model we get the accuracy and precision scores as:**

Accuracy of predictions on Train set: 99.97 %

Accuracy of predictions on Test set: 91.37 %

Precision of predictions on Train set: 100.0 %

Precision of predictions on Test set: 59.17 %

Recall of predictions on Train set: 99.7 %

Recall of predictions on Test set: 36.65 %

f1 score of predictions on Train set: 99.85 %

f1 score of predictions on Test set: 45.26 %

**After resampling when we train our model we get the accuracy and precision scores as**:

Accuracy of predictions on Train set: 99.98 %

Accuracy of predictions on Test set: 92.81 %

Precision of predictions on Train set: 99.98 %

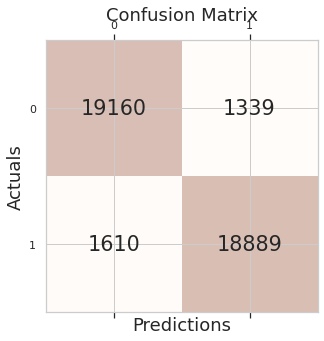
Precision of predictions on Test set: 93.38 %

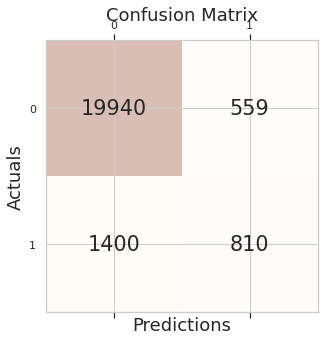
Recall of predictions on Train set: 99.97 %

Recall of predictions on Test set: 92.15 %

f1 score of predictions on Train set: 99.98 %

f1 score of predictions on Test set: 92.76 %



1. KNN:

**Before resampling when we train our model we get the accuracy and precision scores as:**

Accuracy of predictions on Train set: 92.92 %

Accuracy of predictions on Test set: 90.11 %

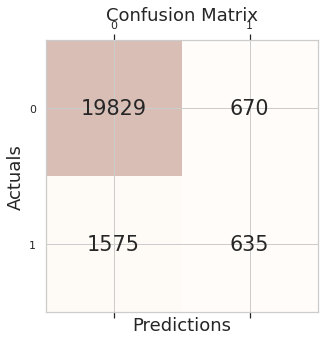
Precision of predictions on Train set: 72.25 %

Precision of predictions on Test set: 48.66 %

Recall of predictions on Train set: 44.21 %

Recall of predictions on Test set: 28.73 %

f1 score of predictions on Train set: 54.85 %

f1 score of predictions on Test set: 36.13 %

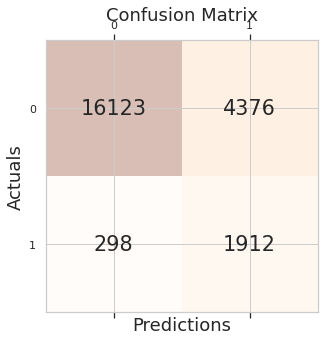
**After resampling when we train our model we get the accuracy and precision scores as**:

Accuracy of predictions on Train set: 88.85 %

Accuracy of predictions on Test set: 85.67 %

Precision of predictions on Train set: 85.35 %

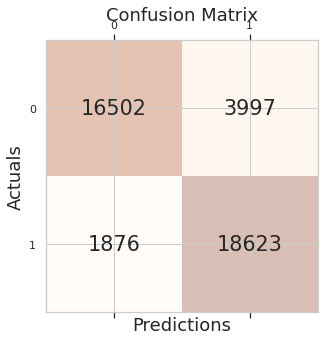
Precision of predictions on Test set: 82.33 %

Recall of predictions on Train set: 93.81 %

Recall of predictions on Test set: 90.85 %

f1 score of predictions on Train set: 89.38 %

f1 score of predictions on Test set: 86.38 %



1. Gaussian Naïve Bayes:

**Before resampling when we train our model we get the accuracy and precision scores as:**

Accuracy of predictions on Train set: 79.11 %

Accuracy of predictions on Test set: 79.42 %

Precision of predictions on Train set: 29.95 %

Precision of predictions on Test set: 30.41 %

Recall of predictions on Train set: 85.67 %

Recall of predictions on Test set: 86.52 %

f1 score of predictions on Train set: 44.39 %

f1 score of predictions on Test set: 45.0 %

**After resampling when we train our model we get the accuracy and precision scores as**:

Accuracy of predictions on Train set: 85.14 %

Accuracy of predictions on Test set: 85.13 %

Precision of predictions on Train set: 78.65 %

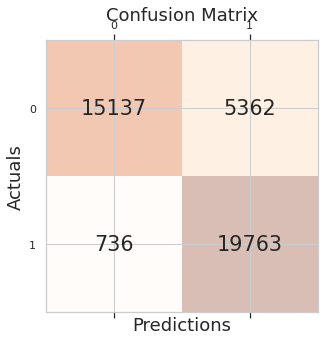
Precision of predictions on Test set: 78.66 %

Recall of predictions on Train set: 96.47 %

Recall of predictions on Test set: 96.41 %

f1 score of predictions on Train set: 86.65 %

f1 score of predictions on Test set: 86.63 %

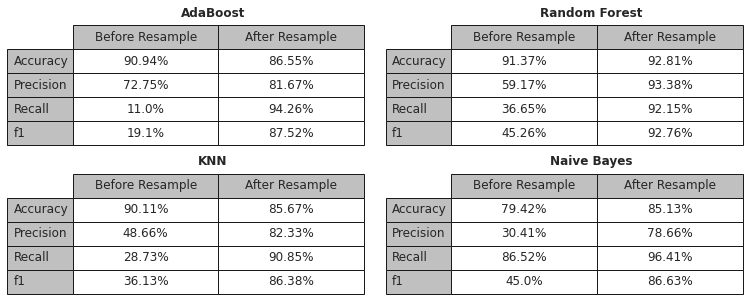


PERFORMANCE VISUALIZATION

At last we generate a 2x2 grid of tables using Matplotlib. Each table contains performance metrics for different machine learning models before and after resampling.

All things considered, our code produces a succinct visual depiction of the performance metrics for several machine learning models both before and after resampling, which facilitates performance comparison.

These nested loops iterate over each subplot in the grid. For each subplot, it sets up the table using the table() method, specifying cell text, row and column labels, cell colors, and other formatting details. It then adds the table to the subplot axes and sets the subplot title.

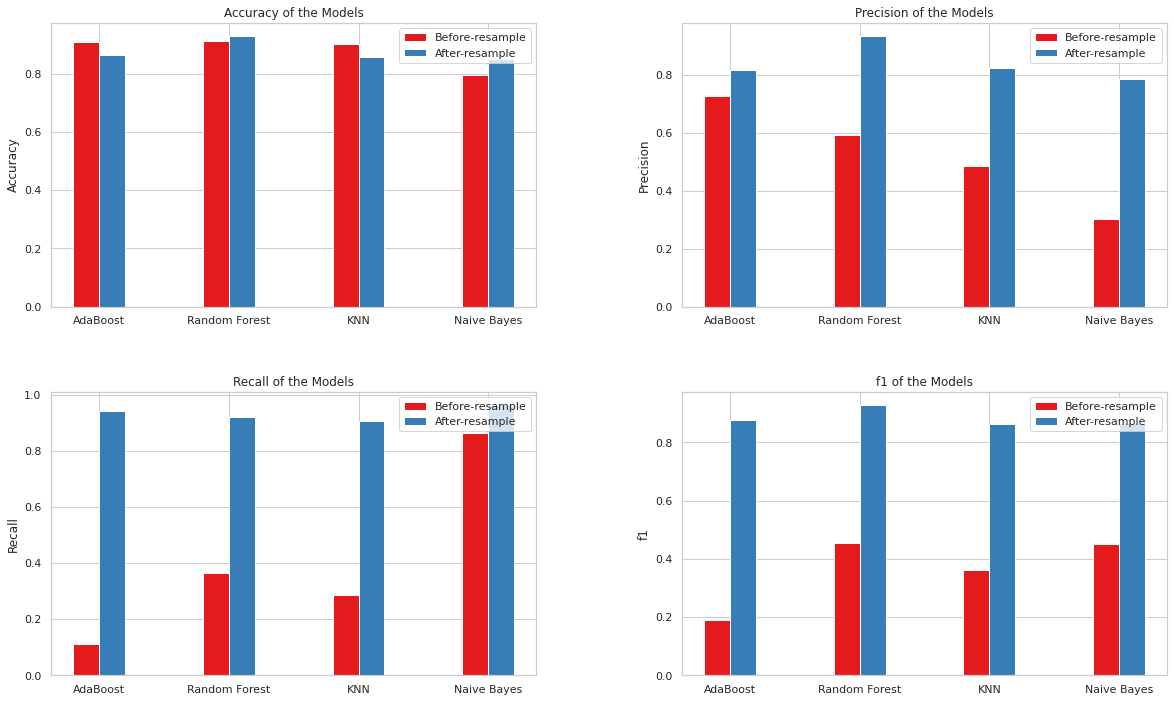


Next our code generates grouped bar charts to visualize the performance metrics (accuracy, precision, recall, and F1-score) of different machine learning models before and after resampling.

The for loop organizes the outcome data for each performance metric (accuracy, precision, recall, and F1-score) into separate lists (dataBarBefore and dataBarAfter). Each list contains the performance metrics for each model, before and after resampling.

It creates a 2x2 grid of subplots with some specified spacing and adjusts the overall size of the figure.

The nested loops iterate over each subplot in the grid. For each subplot, it creates grouped bar charts using the "bar()" function. Each group of bars represents the performance metrics for a particular model, with one set of bars for before resampling and another set for after resampling. It sets the labels, titles, and legends for the plots accordingly.



CONCLUSION:

In conclusion, the application of machine learning (ML) techniques in asteroid threat detection represents a significant advancement in our efforts to protect Earth from potential impact risks. Through the analysis of observational data, feature engineering, and predictive modeling, ML algorithms have demonstrated their effectiveness in enhancing detection, classification, and trajectory prediction capabilities.

Key findings from this report include the successful implementation of ML-based asteroid detection systems, which have significantly improved the efficiency and accuracy of identifying potentially hazardous objects. ML algorithms have enabled the integration of probabilistic modeling, transfer learning, and deep learning architectures, leading to more robust and interpretable detection models.

The importance of ML in advancing asteroid threat detection capabilities cannot be overstated. ML-based systems provide early warning of potential impact threats, allowing for timely mitigation measures to be

implemented. Furthermore, ML techniques contribute to the development of more accurate trajectory forecasts, enabling proactive planning and decision-making in planetary defense efforts.

Looking ahead, future prospects for enhancing planetary defense systems through ML are promising. Continued research and development in ML algorithms, coupled with interdisciplinary collaborations, hold the potential to address persistent challenges such as data scarcity, model interpretability, and computational complexity. By leveraging emerging technologies and fostering collaboration between astronomers, astrophysicists, and machine learning researchers, we can further improve our understanding of asteroid dynamics and enhance our ability to detect and mitigate potential impact risks.

REFERENCES:

1. Mainzer, A. et al. (2011). Preliminary results from NEOWISE: An enhancement to the Wide-field Infrared Survey Explorer for solar system science. The Astrophysical Journal, 731(1), 53.
2. Granvik, M. et al. (2018). De-biased orbit and absolute-magnitude distributions for near-Earth objects. The Astronomical Journal, 155(5), 192.
3. Morbidelli, A. et al. (2012). The orbital distribution of near-Earth objects inside Earth's orbit. Icarus, 219(1), 1-20.
4. Petrillo, C. E. et al. (2020). Asteroid Detection in the 2020s: Opportunities and Challenges. arXiv preprint arXiv:2002.12368.
5. Jump up to Rubin, Alan E.; Grossman, Jeﬀrey N. (January 2010). ”Meteorite and meteoroid:new comprehensive deﬁnitions”. Meteoritic and Planetary Science. 45 (1): 114–122. Bib code:2010MPS...45..114R. doi:10.1111/j.1945-5100.2009.01009.x.
6. Atkinson, Nancy (2 June 2015). What is the diﬀerence between asteroids and meteorites?.Universe Today. Retrieved 13 August 2016.
7. United Nations General Assembly proclaims 30 June as International Asteroid Day. UnitedNations Oﬃce for Outer Space Aﬀairs (Press release). 7 December 2016. UNIS/OS/478.
8. V. Carruba, S. Aljbaae, R. Domingos and W. Barletta, Artificial Neural Network classification of asteroids in the M1:2 mean-motion resonance with Mars, https://doi.org/10.1093/mnras/stab914, arXiv.
9. A. Kumar and A. Malik, “A Comparative Analysis of Various Models for Assessment of Trust in Digital Age Accelerated by Covid-19,” in 2021 International Conference on Computing Sciences (ICCS), 2021, pp. 156–160.
10. R. R. C. Clement et al., “Impact hazard mitigation: understanding the effects of nuclear explosive outputs on comets and asteroids,” in Advanced Maui Optical and Space Surveillance Technologies Conference, 2009, p. E37.
11. V. Carruba, S. Aljbaae, R. C. Domingos, M. Huaman, and W. Barletta, “Machine learning applied to asteroid dynamics,” Celest. Mech. Dyn. Astron., vol. 134, no. 4, p. 36, 2022.
12. C. Prasad, T. A. S. Reddy, B. Kashi, and others, “Asteroid Detection using Machine Learning Algorithm,” Commun. Byurakan Astrophys. Obs., vol. 67, pp. 329–334, 2020.