NASA Hazardous Asteroids Classification

Alison J. March
University of Colorado, Boulder
Boulder, CO 80309-0552
alma2157@colorado.edu

Abstract
The goal of this project is to study and explore effective data mining and Machine Learning methods to identify and classify hazardous asteroids. In addition, we will compare those methods and determine the best machine-learning models for this project.

1. Introduction
The dataset is provided by NASA's Near-Earth Object Observations Program via NeoWs (Near Earth Web Service) Restful API web service for near-earth asteroid information(https://api.nasa.gov/). All the data is maintained by the NASA JPL Asteroid team(http://neo.jpl.nasa.gov/). It consists of 40 data features(columns) and 4,686 data instances (rows) of asteroid entries [1]. We reviewed and pre-processed the dataset to eliminate unnecessary features before using Machine Learning models to complete the classification task.

2. Related Work
So far, we have exported the dataset and done an initial data review and exploratory data analysis with Jupyter Notebook. Redundant features like ***Est Dia in M(min)*** and ***Est Dia in M(max)***, ***Est Dia in Miles(min)***, ***Est Dia in Miles(max)***, ***Est Dia in Feet(min)***, and ***Est Dia in Feet(max)*** were dropped. Furthermore, features like ***Close Approach Date***, ***Orbiting Determination Date***, ***Orbiting Body***, and ***Equinox*** were eliminated before Feature Engineering and Feature Selection. A correlation heatmap is also constructed to study the correlation between data features.

3. Proposed Work
3.1 Dimensionality Reduction
For the remaining 30 data features, we will use the Chi-squared based SelectKBest library to rank each feature based on the correlation score: a high score represents a high correlation and vice versa. We want to drop features with a correlation > 90% to prevent multicollinearity. Min-max scaling normalization will also be carried out to transform the dataset. Finally, we will check and resolve any data imbalance in preparation for Machine Learning algorithms.

3.2 Supervised Machine Learning Models
The dataset is split into training and testing datasets with an 80/20 split ratio.

3.2.1 Logistic Regression
Logistic Regression is usually used to estimate the probability of an instance belonging to a specific class, in this example, we are looking for the probability that this asteroid is hazardous. The mechanism behind it is that if the model predicts that the instance belongs to that class (Hazardous), then it's labeled "1", otherwise it's labeled "0" [2].

The logistic regression model in the vectorized form:

$$\hat{p} = h_\theta(x) = \sigma(x^T\theta)$$

Where $\sigma(\bullet)$ is a sigmoid function that outputs a number between 0 and 1, and $\sigma(t) = \frac{1}{1+\exp(-t)}$ is the logistic function. The code can be completed with *skearn.linear_model.LogisticRegression* class [3].

3.2.2 Decision Trees
Decision Trees (DTs) are a non-parametric supervised learning method that can be used for both classification and regression, with the goal of creating a model that predicts the target variable value by learning simple decision rules inferred from the data features [4]. The process of Decision Trees is as follows: 1) Import *DecisionTreeClassifer* to make DT models. 2) Split the data into training and testing data. 3) Create a Decision Tree model. 4) Put the training data into the model. 5) Predict the model with the values. 6) Check the accuracy of the model's prediction.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad [5]$$

3.2.3 Support Vector Machines
SVMs are particularly suitable for the classification of complex-small or medium-sized datasets, which is perfect for our dataset. We will create the pipeline and hyperparameters and an optimal method for searching for the optimal hyperparameters is to use Scikit-Learn's *GridSearchCV*. Aurélien Géron suggested that when you have no idea what value a hyperparameter should use, a simple approach is to try out consecutive powers of 10 or smaller numbers if a fine-grained search is preferred [6].

3.2.4 Random Forest
Random Forests is an ensemble learning method for both classification and regression as well as other tasks that operate by constructing a multitude of decision trees at training time and is generally trained via the bagging method. We will use the *RandomForestClassifier* class from sklearn.ensemble to train the model and calculate the training and testing accuracy score respectively.

3.2.5 XGBoosting
XGBoosting stands for Extreme Gradient Boosting and is an optimized implementation of Gradient Boosting, and aims to be extremely efficient, flexible, and portable [7]. In recent years, XGBoost has gained significant favor due to the fact it helped

individuals and teams win most of the Kaggle structured data competitions [8]. Due to this reason, I have much more faith in expecting better model results than some of the other machine learning models.

4. Evaluation

We will split the cleaned dataset in the ratio of 80/20 train/test sets. Cross-Validation is a good way to evaluate a model by using Scikit-Learn's cross_val_score() function. Tentatively we will choose the StratifiedKFold class to perform stratified sampling to produce folds that contain a class ratio. The process behind such a method is to create a clone of the classifier at each iteration, train a clone on the training folds and output a prediction on the test fold, and count the number of correct predictions and the ratio of correct predictions. A precision_recall_curve() function can be utilized to visualize the precision and recall versus the decision threshold. The second method to evaluate the model is to compute the confusion matrix, this can be done with Sciki-Learn's cross_val_predict () function. It provides precision and recall scores, and we can combine these two into a single metric called F_1 score.

5. Discussion

This section will be updated once we have trained and evaluated all the models and have stats and insights.

6. Conclusion

This section will be updated once all analyses and results have been collected and documented.

7. References

[1] Shruti Mehta. 2018. NASA: Asteroid classification. (March 2018). Retrieved May 20, 2023, from https://www.kaggle.com/datasets/shrutimehta/nasa-asteroids-classification

[2] Aurélien Géron. 2019. Chapter 4. Training Models. In Hands-on machine learning with SCIKIT-learn, Keras, and TensorFlow: Concepts, tools, and Techniques. SEBASTOPOL: O'REILLY MEDIA, 142–151.

[3] scikit-learn(2009). Sklearn.linear_model.logisticregression. Retrieved May 21, 2023, from https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[4] scikit learn. (2009). 1.10. Decision Trees — scikit-learn 0.22 documentation. Scikit-Learn.org. https://scikit-learn.org/stable/modules/tree.html

[5] Classification: Accuracy. (n.d.). Google for Developers. https://developers.google.com/machine-learning/crash-course/classification/accuracy

[6] Aurélien Géron. 2019. Chapter 2. End-to-End Machine Learning Project. Hands-on machine learning with SCIKIT-learn, Keras, and TensorFlow: Concepts, tools, and Techniques. SEBASTOPOL: O'REILLY MEDIA, 76.

[7] XGBoost Documentation — xgboost 1.7.5 documentation. (n.d.). https://xgboost.readthedocs.io/en/stable/

[8] What is XGBoost? (n.d.). NVIDIA Data Science Glossary. https://www.nvidia.com/en-us/glossary/data-science/xgboost/#:~:text=The%20GPU%2Daccelerated%20XGBoost%20algorithm,dataset%20concurrently%20on%20the%20GPU.