NASA Hazardous Asteroids Classification

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

Abstract

This project aims 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 4,687 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

2.1 Data Pre-processing and Feature Engineering

So far, we have exported the dataset and done an initial data review and exploratory data analysis with Jupyter Notebook. In the process of data preprocessing and cleaning, we did not find any missing data. Furthermore, the ratio of Hazardous versus non-Hazardous Asteroids is 83.89% and 16.11% respectively. In terms of data normalization, we have further categorized object datatypes into date and categorical columns: date_cols = ['Close Approach Date', 'Orbit Determination Date'] and category_cols = ['Orbiting Body', 'Equinox']. For date_cols, both features have been converted to nanoseconds in 64-bit-integer format using the *pd.to_datetime()* conversion. On the other hand, category_cols are transformed using the *OneHotEncoder* method.

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 '***Orbiting Body***', and '***Equinox***' were eliminated before Feature Selection. A correlation heatmap is also constructed to study the correlation between data features.

2.2 Dimensionality Reduction

2.2.1 Feature Selection

For the remaining 32 data features, we used 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, hence the following 6 features have been dropped: ***'Est Dia in M(max)'***, ***'Miles per hour'***, ***'Orbital Period'***, ***'Aphelion Dist'***, ***'Perihelion Time'***, ***'Mean Motion'***. We also filter out constant and quasi-constant features, and duplicated features, this removes the following 7 features: ***'Neo Reference ID'***, ***'Close Approach Date'***, ***'Orbit Determination Date'***,***' Miss Dist.(lunar)'***, ***'Miss Dist.(kilometers)'***, ***'Miss Dist.(miles)'***, ***'Relative Velocity km per hr'***. Finally, we separate the target feature 'Hazardous' from the list of the remaining 19 feature columns.

We then implemented Sequential feature selection algorithms (SFS) using *Mlxend* (machine learning extensions) library to automatically select a subset of features most relevant to this classification problem (see *Figure 1*). From the below graph, we can see that it indicates that Features #3-11 have relevant high scores. This wrapper method is also called naïve sequential feature selection and works very rarely due to the fact it does not account for feature dependence. To verify its accuracy, we will implement another method for comparison [2].

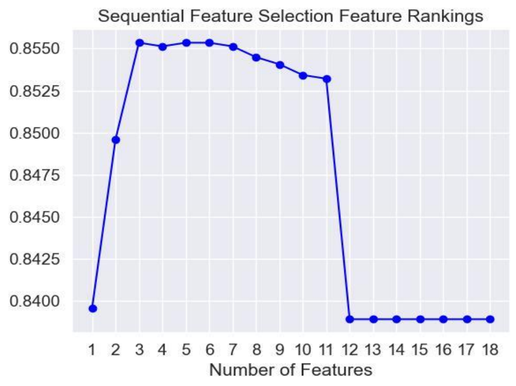


Figure 1

The next algorithm we used is the XGBoost the plot_importance function for feature importance ranking and the result is illustrated in *Figure 2*.

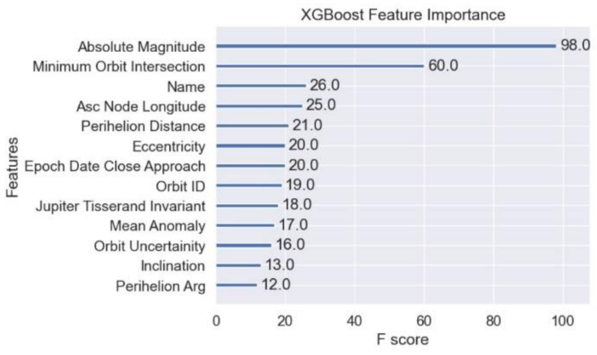


Figure 2

From the above ranking results, we can see that ***'Absolute Magnitude'*** is the most relevant feature followed by ***'Minimum Orbit Intersection'*, *'Orbit Intersection'*, *'Name'*, *'Epoch Date Close Approach'*, *'Epoch Date Close Approach','Orbit ID'*, *'Orbit Uncertainty'*, *'Jupiter Tisserand Invariant'*, *' Perihelion Distance'*, *'Inclination'*, *'Eccentricity'*, *'Asc Node Longitude'*, *'Perihelion Arg'*, *'Mean Anomaly'*, *'Relative Velocity km per sec'*** and ***'Semi Major Axis'***. We can also see why we didn't see ***'Est Dia in M(min)'*** and ***'Epoch Osculation'*** listed in the Feature Importance F1 Score ranking graph, their score is 0 which suggests neither is relevant to predict whether an asteroid is hazardous or not, so we can drop these two features along with ***'Miss Dist.(Astronomical)'***, ***'Semi Major Axis'*** and ***'Relative Velocity km per sec'***. We will be using the remaining 13 predictor features listed in *Figure 2* for Supervised Machine Learning models and evaluations.

2.2.2 Resolving Imbalance

Once the unwanted features have been eliminated, we want to see if the cleaned dataset is balanced or imbalanced since most machine learning algorithms do not work very well with imbalanced datasets. For a classification problem, we can resolve this issue by oversampling the minority class or undersampling the majority class, or simply combining oversampling and undersampling methods to balance the dataset and increase model performance, i.e., by combining *Synthetic Minority Oversampling Technique (SMOTE)* and *Edited Nearest Neighbor (ENN)* method, abbreviated as SMOTE-ENN. Developed by Batista et al (2004) [3], this method combines the SMOTE's ability to generate synthetic minority class examples and ENN's ability to delete some observations from both classes that are having different classes between the observations and its K-nearest neighbor majority class [4]. Before data balancing, we can see that the dataset is imbalanced with 16.11% Hazardous versus 83.89% Non-Hazardous class. After implementing the algorithm, the dataset is more balanced out with 53% versus 47%.

```
Before:
Class=1, Count=755 (16.11%)
Class=0, Count=3932 (83.89%)
```

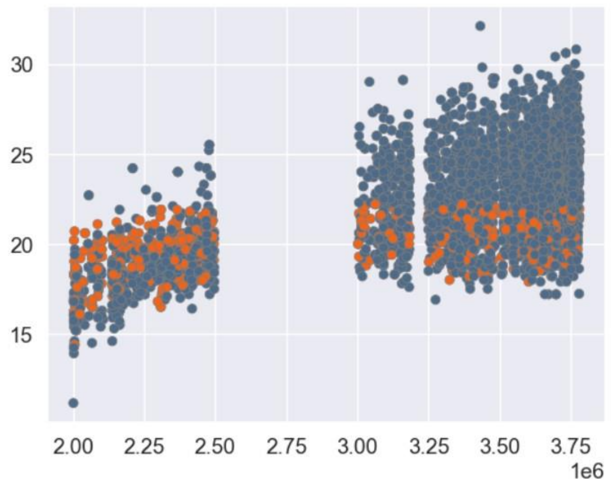


Figure 3. Before the SMOTE+ENN method

```
After:
Class=0, Count=1857 (53.00%)
Class=1, Count=1647 (47.00%)

Text(0.5, 1.0, 'SMOTE+EEN')
```

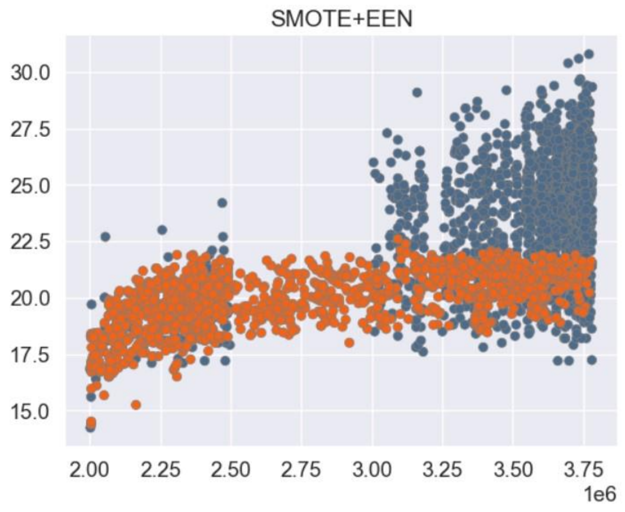


Figure 4. After SMOTE+ENN Method

2.3 Supervised Machine Learning Models

The dataset is split into training and testing datasets with an 80/20 split ratio.

2.3.1 Logistic Regression

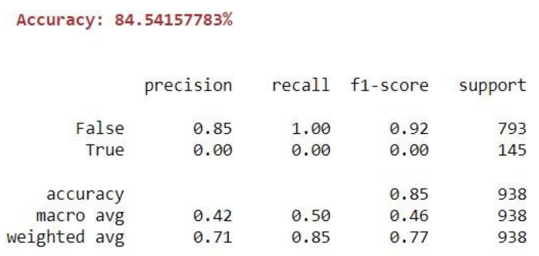
Logistic Regression is usually used to estimate the probability of an instance belonging to a specific class, in this example, we are looking for the probability that this asteroid is hazardous. The mechanism behind it is that if the model predicts that the instance belongs to that class (Hazardous), then it's labeled "1", otherwise it's labeled "0" [5].

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 [6]. We noticed that before and after the SMOTE-EEN method to balance the dataset, the accuracy score for the Logistic Regression went down from 84.54% to 53.5%. This is normal because SMOTE technique creates synthetic data points based on the original data points and does not copy those data points but creates slightly varied ones. The dataset is balanced so it will predict the minority classes better, but the overall accuracy will decrease.

Furthermore, the most crucial aspect of this result is that we should not rely on the Accuracy score as the parameter to determine the performance. Instead, we can try the F_1 score, AUC, Classification Report containing Recall, and Precision to determine if the model has been improved or not. Figure 5 displays the overall performance metrics for the logistic model.



```
Accuracy: 84.54157783%

              precision    recall  f1-score   support

       False       0.85      1.00      0.92       793
        True       0.00      0.00      0.00       145

    accuracy                           0.85       938
   macro avg       0.42      0.50      0.46       938
weighted avg       0.71      0.85      0.77       938
```

Figure 5. Metrics Report for Logistic Regression Model

2.3.2 *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 [7]. 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, where Accuracy= $\frac{TP+TN}{TP+TN+FP+FN}$ [8]

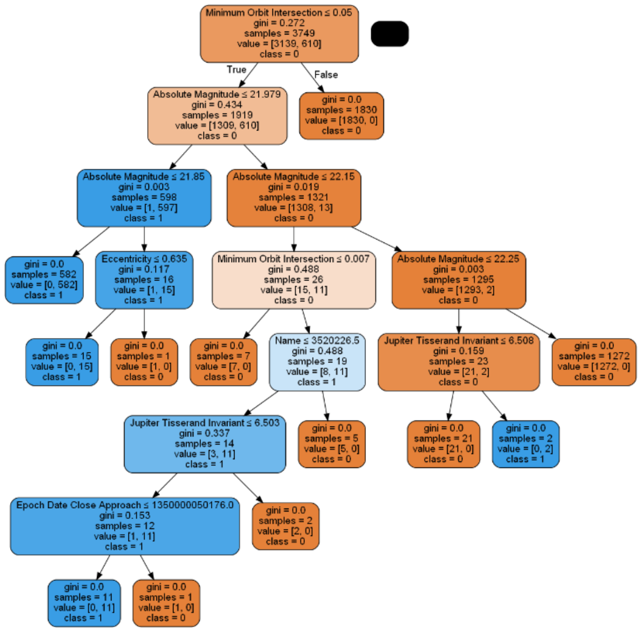


Figure 6. The Decision Tree Graph

The calculated Accuracy score is 99.47%, ROC AUC is 98.28%, it also gives the feature importance ranking (shown below).

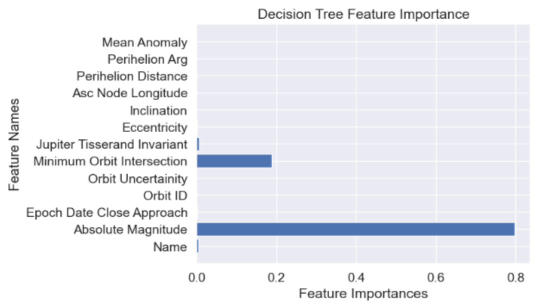
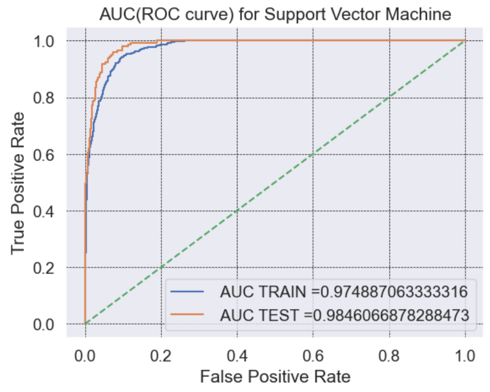


Figure 7

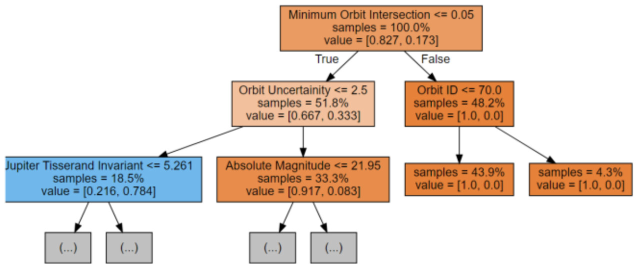
2.3.3 *Support Vector Machines*

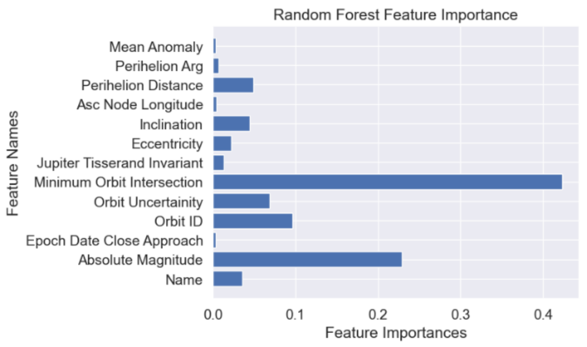
SVMs are particularly suitable for the classification of complex-small or medium-sized datasets, which is perfect for our dataset. We will create a pipeline and hyperparameters and an optimal method for searching for the optimal hyperparameters is to use Scikit-Learn's *GridSearchCV*. Aurélien Géron 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 [9]. We used the linear kernel and set the gamma parameter to "auto". The calculated accuracy score is 95.10%, and the ROC AUC score is 98.69%.



2.3.4 *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 scores respectively. With parameters of 500 estimators, 18 leaf nodes, our accuracy is 99.04%, and ROC AUC score is 99.96%.

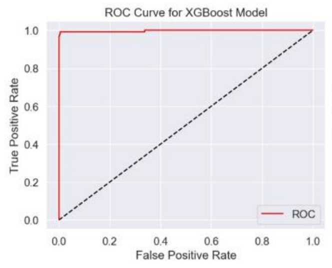


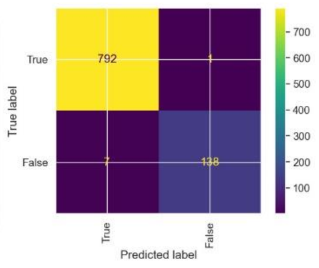


2.3.5 *XGBoost*

XGBoosting stands for Extreme Gradient Boosting and is an optimized implementation of Gradient Boosting, and aims to be extremely efficient, flexible, and portable [10]. 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 [11]. Due to this reason, I have much more faith in expecting better model results than some of the other machine learning models.

The Accuracy score for XGBoost model is 99.147% and ROC AUC score is 99.76%.





3. Proposed Work

3.1 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.

We plan to compare all the machine models' performance metrics and plot their performance scores in a single graph. In addition, we will double-check whether overfitting or underfitting occurred, and quality check on data pipelines.

Lastly, TensorFlow will be used for learning purposes to perform binary classification on the NASA Asteroid dataset and compare the results with the traditional ml models.

4. Discussion

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

5. Conclusion

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

6. 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] https://analyticsindiamag.com/a-complete-guide-to-sequential-feature-selection/

[3] Batista, G. E., Prati, R. C., &amp; Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter, 6(1), 20–29. https://doi.org/10.1145/1007730.1007735

[4] Viadinugroho, R. A. A. (2021, September 30). Imbalanced classification in Python: Smote-enn method. Medium. https://towardsdatascience.com/imbalanced-classification-in-python-smote-enn-method-db5db06b8d50

[5] 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.

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

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

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

[9] 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.

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

[11] 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.