DATS 6501 Capstone: NASA: Detecting Potentially Hazardous Asteroids

Instructor: Dr. Abdi Awl

Final Report

Group 3 Members: Rayna Liu, Zheyue Wang

Date: 4/27/2022

Table of Contents:

1. Glossary of Terms
2. Introduction
    - 2.1 Introduction/Background
    - 2.2 Problem Statement
    - 2.3 Problem Elaboration
    - 2.4 Motivation
    - 2.5 Project Scope
3. Literature Review
    - 3.1 Relevant Research
4. Methodology
    - 4.1 Dataset Description
    - 4.2 Data Collection
    - 4.3 Data Inspection & Wrangling
    - 4.4 Exploratory Data Analysis
    - 4.5 Data Preprocessing and/or Feature Engineering
    - 4.6 Data Modeling & Visualizations
5. Results & Analysis
6. Conclusion
    - 6.1 Conclusion
    - 6.2 Project Limitation
    - 6.3 Future Research
7. References
8. Appendix

1. Glossary of Terms:

More glossary of terms in website link - **JPL - Glossary**

- ***PHA*:** Potentially Hazardous Asteroids are currently defined based on parameters that measure the asteroid's potential to make threatening close approaches to the Earth.

- ***NEO***: A near-Earth Object is an asteroid or comet which passes close to the Earth's orbit. In technical terms, a NEO is considered to have a trajectory that brings it within 1.3 astronomical units of the Sun and hence within 0.3 astronomical units, or approximately 45 million kilometers, of the Earth's orbit.

- ***SPK***: S(spacecraft) and P(planet) Kernel. Historically, ephemerides for spacecraft have been organized differently from planets and satellites. They are usually generated through different processes and using different representations. However, there is no essential reason for keeping them separate. A spacecraft, planet, satellite, comet, or asteroid has a position and velocity relative to some center of mass and reference frame.

- ***Equinox***: An equinox is traditionally defined as the time when the plane of Earth's equator passes through the geometric center of the Sun's disk.

2. Introduction

2.1 Introduction/Background:

Asteroids, sometimes called minor planets , are small rocky bodies that orbit the Sun. The currently known asteroid count is 1,113,527 [1]. Most asteroids are within the main asteroid belt between Mars and Jupiter, some have more eccentric orbits, and a few pass close to the Earth or enter the atmosphere as meteors.

Scientists continuously monitor asteroids, especially potentially hazardous asteroids, through observatory observation and launching probes. Potentially hazardous asteroids, whose orbit can make close approaches to the Earth and are large enough to cause significant regional damage in the event of an impact.

2.2 Problem Statement:

Asteroids are concerned that they will get gravitational disturbances that modify their orbits as they orbit the Sun. Some of them come close to Earth, known as near-Earth objects. Since they are at risk of hitting Earth, it is important to build machine learning models to detect potentially hazardous asteroids that could help spot the danger early and respond to it in time.

2.3 Problem Elaboration:

2.3.1 Data Inspection & Wrangling:

In this section, we cleaned data before doing EDA, checked duplicate observations, explored the columns of the DataFrame, analyzed them accordingly, handled missing values, and converted column data types.

2.3.2 Exploratory Data Analysis:

We analyzed and visualized the categorical variables to understand their distribution in this section. The dataset contains eight categorical variables: *id*, *spkid*, *full_name*, *neo*, *pha*, *orbit_id*, *equinox*, and *class*. We came up with two questions: What is the distribution of near-earth objects, potential hazards, asteroids, and orbit classification, respectively? What relationship is between near-earth objects, potential hazards asteroids, and orbit classification?

2.3.3 Data Preprocessing:

For this section, we processed our data in the following steps: handle identifiers, encode categorical features and target, split train, validation, test sets, normalize the data to make sure all the numeric features are on the same scale, and handle class imbalance.

2.3.4 Models:

We turned hyperparameters of eight models, including Random Forest, Extreme Gradient Boosting (XGBoost), Decision Tree, Naive Bayes, Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbor, and one Neural Network model, Multilayer Perceptrons Model. We compared these eight models and selected the best model, a Multilayer Perceptron model. Then, we did visualization on the best model, and we plotted the training history, loss history, and confusion matrix.

2.4 Motivation:

This project was inspired by the movie *Don't Look Up*, which tells the story of a comet above five miles wide heading towards Earth with only six and a half months of advance warning. This brings us to the science of asteroid defense. "NASA simulation revealed that six months' warning isn't enough to stop an asteroid from hitting Earth. We'd need 5 to 10 years" [2]. The idea is not to find these potentially hazardous asteroids when they're just a few days away from impact but to find them when they're decades away from the ideal impact so that we have time to devise ways to prevent the devastating effects.

2.5 Project Scope:

This project applies data science to the field of aerospace exploration to witness the power of machine learning on massive datasets. This project is based on January 31, 2022, asteroid data collected by the Jet Propulsion Laboratory of California Institute of Technology to develop multiple intelligent machine algorithm models to detect hazardous potential asteroids. In the end, finding an optimal model could help NASA scientists spot potentially hazardous asteroids early and respond to them in time to mitigate the damage from a devastating impact.

3. Literature Review:

3.1 Relevant Research:

Hefele, J. D., Bortolussi, F., & Zwart, S. P. (2020). Identifying Earth-impacting asteroids using an artificial neural network. *Astronomy & Astrophysics*, *634*. https://doi.org/10.1051/0004-6361/201935983
https://www.aanda.org/articles/aa/full_html/2020/02/aa35983-19/aa35983-19.html
For this paper, the author has used the MLP model to predict PHA and adjust the model to get better performance. In this article, the author used five features as input, with two hidden layers and one output.

Si, A. S. (2020). Hazardous Asteroid Classification through Various Machine Learning Techniques. *International Research Journal of Engineering and Technology (IRJET)*, *07*(03), 5388–5390
https://trello.com/c/ueztaNgD/73-hazardous-asteroid-classification-through-various-machine-learning-techniques
In this paper, the author has used eight machine learning techniques to classify the 4,688 asteroids into hazardous and non-hazardous. Of them, random forest (with the number of trees is 15) and XGBoost methods give the highest accuracy of 100%.

Blair, N., & Masiero, J. (2019, August 15). *Deep Neural Networks for Near-Earth Object Classification in Wise Data*.
https://trello.com/c/FQAT9Xbs/91-blair-n-masiero-j-2019-august-15-deep-neural-networks-for-near-earth-object-classification-in-wise-data
This document mentioned the application of multi-layer perceptrons for Near-Earth Object Classification. The best MLP model achieved 95% accuracy. This model had a depth of seven (one input, two hidden, one output) and three batch normalization layers after input and each hidden layer. A hidden width of 100 neurons.

4. Methodology

4.1 Dataset Description:

This data is taken from Kaggle. The uploader collected this dataset officially maintained by the Jet Propulsion Laboratory of California Institute of Technology, an organization under NASA. This dataset includes 958,524 unique asteroids, 44 attributes which contain 7 identify attributes, 4 size attributes, 33 orbital attributes, and one target variable (potential hazards asteroid or not).

	id	spkid	full_name	pdes	name	prefix	neo	pha	H	diameter	...	sigma_i	sigma_om	sigma_w	sigma_ma	sigma_ad	sigma_n	sigma_tp
0	a0000001	2000001	1 Ceres	1	Ceres	NaN	N	N	3.40	939.400	...	4.608900e-09	6.168800e-08	6.624800e-08	7.820700e-09	1.111300e-11	1.196500e-12	3.782900e-08
1	a0000002	2000002	2 Pallas	2	Pallas	NaN	N	N	4.20	545.000	...	3.469400e-06	6.272400e-06	9.128200e-06	8.859100e-06	4.961300e-09	4.653600e-10	4.078700e-05
2	a0000003	2000003	3 Juno	3	Juno	NaN	N	N	5.33	246.596	...	3.223100e-06	1.664600e-05	1.772100e-05	8.110400e-06	4.363900e-09	4.413400e-10	3.528800e-05
3	a0000004	2000004	4 Vesta	4	Vesta	NaN	N	N	3.00	525.400	...	2.170600e-07	3.880800e-07	1.789300e-07	1.206800e-06	1.648600e-09	2.612500e-10	4.103700e-06
4	a0000005	2000005	5 Astraea	5	Astraea	NaN	N	N	6.90	106.699	...	2.740800e-06	2.894900e-05	2.984200e-05	8.303800e-06	4.729000e-09	5.522700e-10	3.474300e-05
5	a0000006	2000006	6 Hebe	6	Hebe	NaN	N	N	5.80	185.180	...	2.191800e-06	1.122100e-05	1.300600e-05	7.392200e-06	3.306700e-09	4.438800e-10	2.875400e-05

Figure 1: *The first five rows of the raw asteroid dataset*

The basic definitions of the columns have been given below and more detail are on the website link about Jet Propulsion Laboratory of California Institute of Technology JPL Small-Body Database Search Engine.

Columns Definition:

Identify Variables:

id: object internal database id
spkid: object primary SPK-ID, SPK stands for "Spacecraft and Planet Kernel"
full_name: object full name/designation
pdes: serial
name: object name
prefix: object prefix
neo: near-earth object or not (Y/N)
pha: potential hazards asteroid or not (Y/N)

Asteroid Size Variables:

H: absolute magnitude parameter, is the visual magnitude an observer would record if the asteroid were placed 1 Astronomical Unit (au) away, and 1 au from the Sun and at zero phase angle. In the other word, H is the magnitude of an asteroid at zero phase angle and at unit heliocentric and geocentric distances

diameter: asteroid diameter (from an equivalent sphere) (km)
albedo: geometric albedo, is the ratio of a body's brightness at zero phase angle to the brightness of a perfectly diffusing disk with the same position and apparent size as the body. The albedo combined with the absolute magnitude can help determine the size of an asteroid
diameter_sigma: 1-sigma (68%) uncertainty in object diameter (km)

Orbital Variables:
orbit_id: orbit solution ID
epoch: the epoch of osculation in Julian day form (TDB)
epoch_mjd: the epoch of osculation in modified Julian day form (TDB)
epoch_cal: epoch of osculation in calendar date/time form (TDB)
equinox: equinox of reference frame
e: eccentricity
a: semi-major axis (au), is the longest semidiameter or one-half of the major axis
q: perihelion distance (au), an orbit's closest point to the Sun
i: inclination, angle with respect to the x-y ecliptic plane (deg), the angle between the orbit plane and the ecliptic plane
om: omega, the longitude of ascending node, is the angle from a specified reference direction, called the origin of longitude, to the direction of the ascending node, as measured in a specified reference plane.
w: argument of periapsis, is the angle from the body's ascending node to its periapsis, measured in the direction of motion
ma: mean anomaly (deg)
ad: aphelion distance (au), an orbit's farthest point to the Sun
n: mean motion (deg/d)
tp: time of perihelion passage (TDB)
tp_cal: time of perihelion passage in calendar date/time (TDB)
per: sidereal orbital period (d)
per_y: sidereal orbital period year (years)
moid: earth minimum orbit intersection distance (au)
moid_ld: earth minimum orbit intersection distance (LD)
sigma_e: 1-sigma uncertainty eccentricity
sigma_a: 1-sigma uncertainty semi-major axis (au)
sigma_q: 1-sigma uncertainty perihelion distance (au)
sigma_i: 1-sigma uncertainty inclination (deg)
sigma_om: 1-sigma uncertainty longitude of ascending node
sigma_w: 1-sigma uncertainty argument of periapsis
sigma_ma: 1-sigma uncertainty mean anomaly (deg)
sigma_ad: 1-sigma uncertainty aphelion distance (au)

sigma_n: 1-sigma uncertainty mean motion (deg/d)
sigma_tp: 1-sigma uncertainty time of perihelion passage (TDB)
sigma_per: 1-sigma uncertainty sidereal orbital period (d)
class: orbit class
rms: root mean square of the signal

Units Definition:
au: astronomical unit, the astronomical unit (au) is defined by the IAU as exactly 149,597,870,700 m
LD: lunar distance refers to the average distance between the Earth and Moon

4.2 Data Collection:

This asteroid data was collected by the Jet Propulsion Laboratory of the California Institute of Technology on January 31, 2022. Here is the link - Asteroid Dataset.

4.3 Data Inspections & Wrangling:

4.3.1 Check Duplicate Observations:

```
# Check duplicate observations
pd.unique(raw_data["id"]).shape
```

```
(958524,)
```

Figure 2: *Check duplicate observations*

Since the *id* represents the unique number of the asteroid, which match the record of the total number row, thus, our data doesn't contain duplicate observations.

4.3.2 Handle Missing Values:

```
# Check the null value
raw_data.isnull().sum()
```

```
id                     0
spkid                  0
full_name              0
pdes                   0
name              936460
prefix            958506
neo                    4
pha                19921
H                   6263
diameter          822315
albedo            823421
diameter_sigma    822443
orbit_id               0
epoch                  0
epoch_mjd              0
epoch_cal              0
equinox                0
e                      0
a                      0
q                      0
i                      0
om                     0
w                      0
ma                     1
ad                     4
n                      0
tp                     0
tp_cal                 0
per                    4
per_y                  1
moid               19921
moid_ld              127
sigma_e            19922
sigma_a            19922
sigma_q            19922
sigma_i            19922
sigma_om           19922
sigma_w            19922
sigma_ma           19922
sigma_ad           19926
sigma_n            19922
sigma_tp           19922
sigma_per          19926
class                  0
rms                    2
dtype: int64
```

Figure 3: *The number of null values in each column*

We explored the columns of the raw dataset and analyzed them accordingly. We removed the columns that cover 85% of them are null and do not contribute to EDA or modeling. 97.7% of the *name* column is null. Since the *full_name* column includes *pdes* and *name* and has no null, *pdes* and *name* columns could be removed. *prefix* refers to asteroid prefix, and 99.9% of this column is null. It could be removed because it does not contribute to EDA or modeling. About 86% null in each *diameter*, *albedo*, and *diameter_sigma* column. These attributes describe the asteroid's size, and *H*, which refers to the absolute magnitude parameter, could also tell the size of the asteroid. Additional values in these three columns are specific to each unique asteroid, and replacing the mean or median will lead to inaccurate subsequent analysis and model construction. Thus, remove the *diameter*, *albedo*, and *diameter_sigma* column. Other columns also have null values that account for up to 2.1% of their columns. Therefore, keep these columns, and the rows that include null values could be removed.

After handling the missing value, the dimensionality of no null dataset is (932335, 39).

4.3.3 Convert Column Data Type:

```
# Check the information of no null dataset
no_null_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 932335 entries, 0 to 932334
Data columns (total 39 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   id         932335 non-null  object
 1   spkid      932335 non-null  int64
 2   full_name  932335 non-null  object
 3   neo        932335 non-null  object
 4   pha        932335 non-null  object
 5   H          932335 non-null  float64
 6   orbit_id   932335 non-null  object
 7   epoch      932335 non-null  float64
 8   epoch_mjd  932335 non-null  int64
 9   epoch_cal  932335 non-null  float64
 10  equinox    932335 non-null  object
 11  e          932335 non-null  float64
 12  a          932335 non-null  float64
 13  q          932335 non-null  float64
 14  i          932335 non-null  float64
 15  om         932335 non-null  float64
 16  w          932335 non-null  float64
 17  ma         932335 non-null  float64
 18  ad         932335 non-null  float64
 19  n          932335 non-null  float64
 20  tp         932335 non-null  float64
 21  tp_cal     932335 non-null  float64
 22  per        932335 non-null  float64
 23  per_y      932335 non-null  float64
 24  moid       932335 non-null  float64
 25  moid_ld    932335 non-null  float64
 26  sigma_e    932335 non-null  float64
 27  sigma_a    932335 non-null  float64
 28  sigma_q    932335 non-null  float64
 29  sigma_i    932335 non-null  float64
 30  sigma_om   932335 non-null  float64
 31  sigma_w    932335 non-null  float64
 32  sigma_ma   932335 non-null  float64
 33  sigma_ad   932335 non-null  float64
 34  sigma_n    932335 non-null  float64
 35  sigma_tp   932335 non-null  float64
 36  sigma_per  932335 non-null  float64
 37  class      932335 non-null  object
 38  rms        932335 non-null  float64
dtypes: float64(30), int64(2), object(7)
memory usage: 277.4+ MB
```

Figure 4: *The data type of each column of the no null dataset*

When we checked the information of the new dataset, we found that *spkid* refers to object primary SPK-ID. It should be converted from int64 to object. Next, we saved the new dataset in a CSV file for preparing EDA in Tableau.

4.4 Exploratory Data Analysis:

The interactive visualization is in Tableau Public. Here is the link.

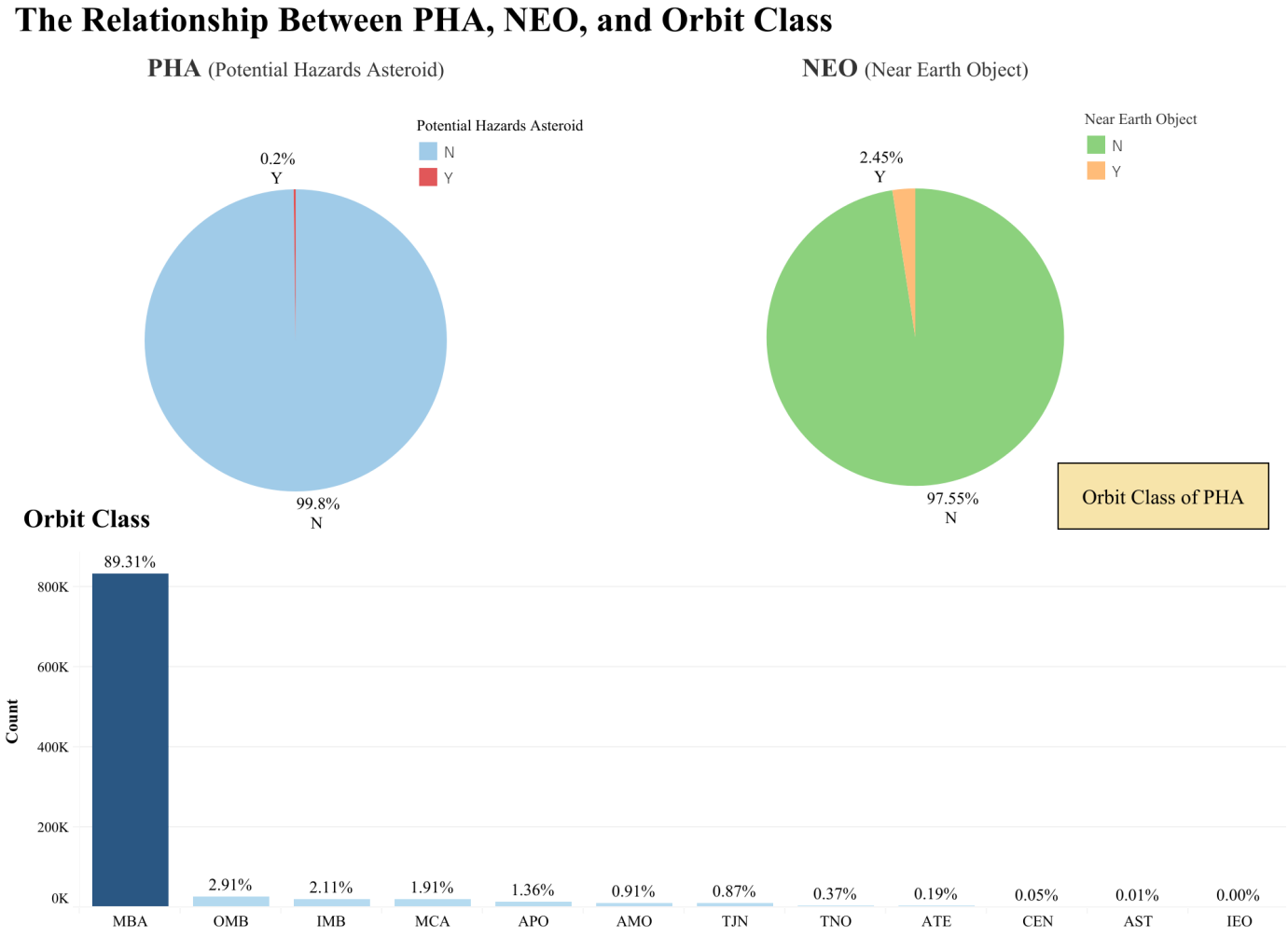

Figure 5: *The relationship between PHA, NEO, and Orbit Class*

The figure above shows the overall EDA of our dataset. Of these 932,335 asteroids, 97.55% are non-near-earth objects, and 2.45% are near-earth objects. 99.8% of them are safe asteroids, and 0.2% are hazardous asteroids. Our data is highly imbalanced in the target variable as almost all data are concentrated on non-hazardous asteroids. There are 12 orbit classes. Most of the orbit class of asteroids is MBA, which is 89.31%.

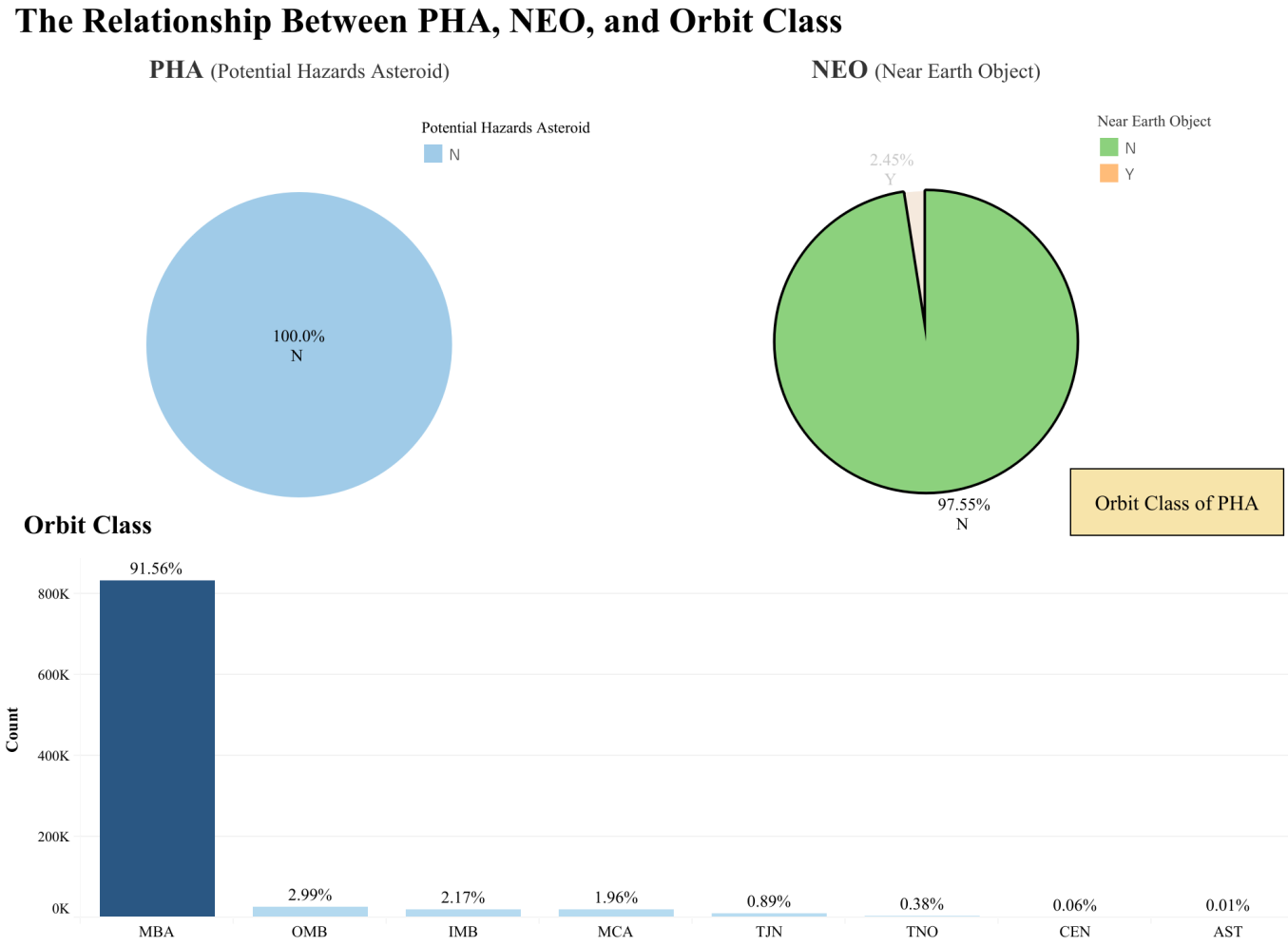

Figure 6: *What distribution of PHA and Orbit Class in Non-Near-Earth Object?*

From Figure 5 above, all asteroids are non-near-earth objects that are also non-potentially hazardous. Most of them have orbit class in MBA.

The Relationship Between PHA, NEO, and Orbit Class

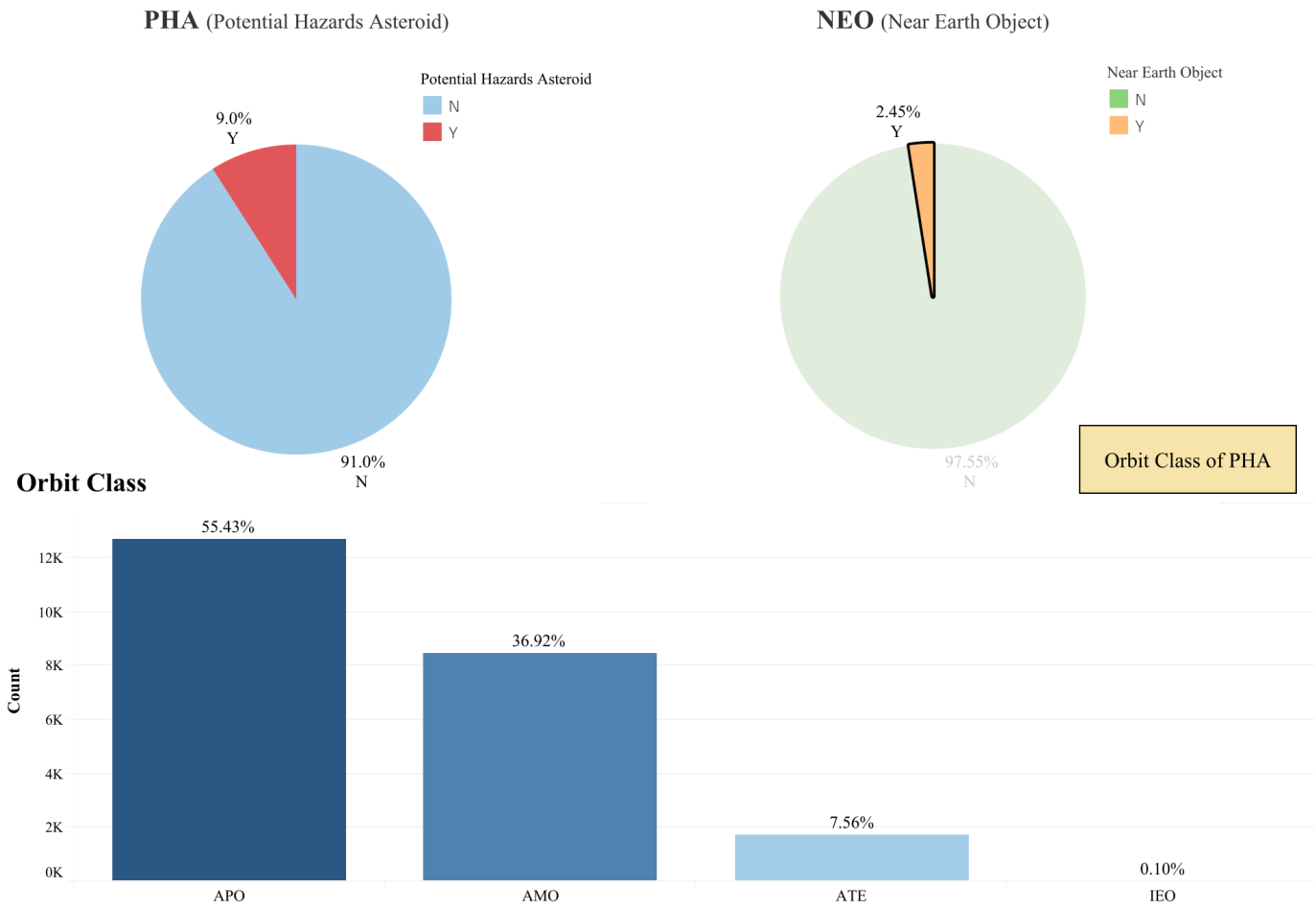

Figure 7: *What distribution of PHA and Orbit Class in Near-Earth Object?*

Of the Near-Earth Objects, 91% of them are safe, and 9% of them are hazardous. The orbit classes of Near-Earth Objects are APO, AMO, ATE, and IEO.

Of the near-earth objects, 9% of them are potentially hazardous asteroids. Their orbit class is Apollo, Aten, Amor, and Interior-Earth Objects.

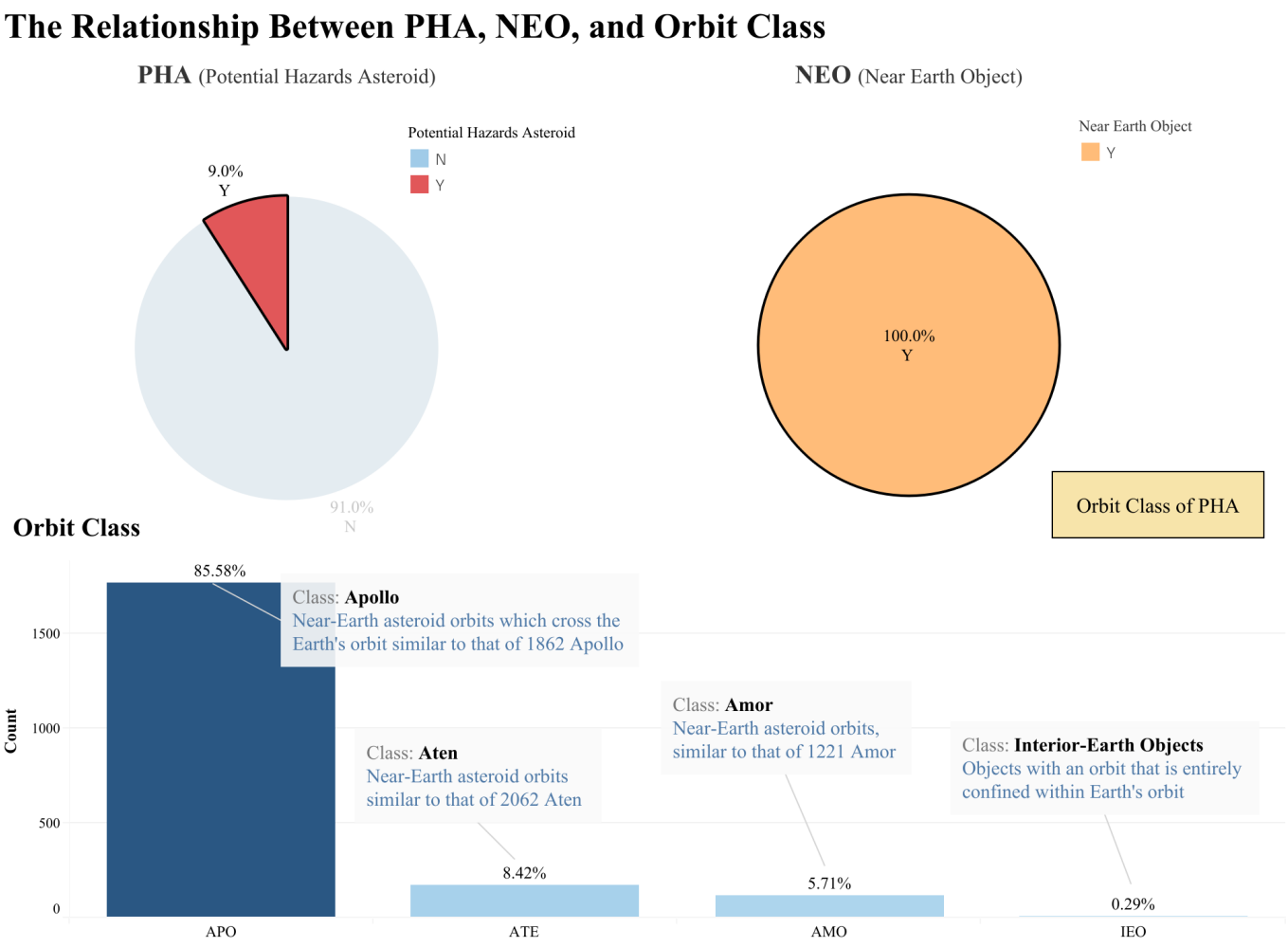

Figure 8: *What distribution of Orbit Class in Potential Hazardous Asteroid?*

Most of the potentially hazardous asteroids have orbit class in Apollo, which is 85.58%. 8.42% of them have orbit class in Aten, 5.71% have orbit class in Amor, and 0.29% have orbit in Interior-Earth Objects.

4.5 Data Preprocessing and/or Feature Engineering:

4.5.1 Handel Identifiers and Encode Features and Target:

The identifier would not contribute to modeling, so we removed these identifying columns: *id*, *spkid*, *full_name*, *orbit_id*, and *equinox*. We encode categorical features *neo* and *class*; we created a new variable for each level of a categorical feature. We also encoded categorical target *pha*, and 0 represents the safe asteroid, and 1 represents the hazardous asteroid.

4.5.2 Split Train, Validation, and Test Set in the ratio 60:20:20:

"If the size of our dataset is between 100 to 1,000,000, then we split it in the ratio 60:20:20" [3]. Since our data is large enough (932,335 unique asteroids), we split data into three parts: train dataset (used to train the model), validation dataset (evaluated the performance of the model and choose the best model), and test dataset (final measured the performance of the model that we choose) to test our model performance. We used stratify parameter to split so that the proportion of *pha* in the train, validation, and test set produced will be the same as the proportion of *pha* provided to the whole dataset.

```
Shape of original dataset : (932335, 46)
Shape of x_train set (559401, 45)
Shape of y_train set (559401, 1)
Shape of x_validation set (186467, 45)
Shape of y_validation set (186467, 1)
Shape of x_test set (186467, 45)
Shape of y_test set (186467, 1)
```

Figure 9: *The dimensionality of train, validation, and test set*

4.5.3 Normalizing:

We used StandardScaler to normalize the features in the train, validation, and test set to make sure all the numeric features are on the same scale. We chose *StandardScaler* because it is useful in classification, and outliers have little influence when normalizing. (*Normalizer* is useful in regression, and *MinMaxScaler* is sensitive to outliers). Also, normalizing after splitting could prevent leaking information about the validation and test set into the train set.

4.5.4 Handle Class Imbalance:

From Figure 10 below, we saw that the target variable *pha* is highly imbalanced in the train set, 99.778% of asteroids are non-hazardous, and 0.221% of asteroids are hazardous. Since the amount of our minority class is extremely small, we had to handle the data imbalance issue. The traditional method for solving this issue is

oversampling or undersampling data. For dealing with the over 10,000 data, if we simply over-sampled our data, our machine wouldn't be able to run that as it overloads. Besides, only using an over-sample or an under-sample will cause overfitting or under-fitting problems in our model. Thus, in this case, we combined upsampling and undersampling, which is upsampling the minority class by using SMOTE to synthesize the minority class, to let the minority class sample be the half of the majority class sample after resampling. We used random undersampling to downsample the majority class to let the number of majority class and minority class be balanced after resampling.

```
# Imbalance in target variable
y_train.value_counts()
```

```
pha
0      558161
1        1240
dtype: int64
```

```
# Data Upsampling - SMOTE
x_train_us, y_train_us=SMOTE(sampling_strateg
y_train_us.value_counts()
```

```
pha
0      558161
1      279080
dtype: int64
```

```
# Data Undersampling - Random Undersampling
random_under_sampling=RandomUnderSampler(rand
x_train_us_rus, y_train_us_rus=random_under_s
y_train_us_rus.value_counts()
```

```
pha
1      279080
0      279080
dtype: int64
```

Figure 10: *Imbalance in target variable (Before)*
Balance in target variable (After Upsampling + Random Under Sampling)

From Figure 10 above, we set *sampling_strategy*=0.5 in Upsampling SMOTE, which means the minority class (pha=1) was upsampled to 50% the number of asteroids of the majority class (pha=0). We chose a rate of 0.5 for two reasons.

The first reason is that we did not want to generate as many minority classes as majority classes so that our train data would contain 1,116,322 data, and the time of model running and hyperparameter turning would excessively increase. The second reason is that 0.5 is more suitable for our data size, model speed, and model performance. A low rate (less than 0.5) led to the bad performance of the model to classify minority classes, while a high rate (greater than 0.5) led to a long time for the model to run and turn parameters and the ability to classify the minority classes has not increased.

After upsampling, we randomly under-sampled the number of majority class (pha=0) as the number of minority class (pha=1). The number of majority classes equals the number of the minority class in the training dataset after handling class imbalance.

4.5.5 Data Modeling & Visualizations:

We used seven machine learning models and one multilayer perceptron model based on our literature reviews. With binary classification, it is very intuitive to score the model in terms of scoring metrics such as precision, recall, and f1-score. We decided to use macro-averaging f1-score as metrics to evaluate the model's efficiency because all classes need to be treated equally to evaluate the classifier's overall performance concerning the most frequent class labels.

A. Machine Learning Models:

A.1 <u>Random Forest</u> (Literature Review Recommendation)

Random forest used to solve regression and classification problems consists of many decision trees. A random forest eradicates the limitations of the decision tree algorithm. It reduces the overfitting of the dataset and increases precision. It is used for the dataset with high dimensionality and does not need to make feature selection before modeling.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	186054
1	0.66	0.97	0.79	413
accuracy			1.00	186467
macro avg	0.83	0.98	0.89	186467
weighted avg	1.00	1.00	1.00	186467

Duration: 0:02:45.905846

Figure 11: *Metrics of the Random Forest in Validation Set*

A.2 <u>XGBoost</u> (Literature Review Recommendation)
XGBoost stands for Extreme Gradient Boosting, which is an additive and sequential model where trees are grown in a sequential manner that converts weak learners into strong learners by adding weights to the weak learners and reducing the weights of the strong learners. So each tree learns and boosts from the previous tree grown. XGBoost is fast in training large datasets compared to other gradient boosting implementations. It has higher predicting power and performance and is commonly used and frequently makes its way to the top of the leaderboard of competitions in data science.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	186054
1	0.66	0.99	0.79	413
accuracy			1.00	186467
macro avg	0.83	0.99	0.90	186467
weighted avg	1.00	1.00	1.00	186467

Duration: 0:06:08.366335

Figure 12: *Metrics of the XGBoost in Validation Set*

A.3 <u>Decision Tree</u>
A decision tree is used to categorize or predict based on how a previous set of questions were answered. It is a robust model able to deal with class imbalance data. Also, it is simple to understand and interpret what features contribute to classification performance on safe and dangerous asteroids.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	186054
1	0.67	0.97	0.79	413
accuracy			1.00	186467
macro avg	0.84	0.98	0.90	186467
weighted avg	1.00	1.00	1.00	186467

Duration: 0:00:07.427345

Figure 13: *Metrics of the Decision Tree in Validation Set*

A.4 Naive Bayes

Naive Bayes is used to solve classification problems. For numerous data with many features, the best solution would be to use the Naive Bayes classifier, which is quite faster in comparison to other classification algorithms.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	186054
1	0.00	0.00	0.00	413
accuracy			1.00	186467
macro avg	0.50	0.50	0.50	186467
weighted avg	1.00	1.00	1.00	186467

Duration: 0:00:00.985060

Figure 14: *Metrics of the Navie Bayes in Validation Set*

From Figure 14 above, we found that Naive Bayes works poorly because it works well when all features are independent. In our dataset, the features are correlative. Thus Naive Bayes is broken and predicted all values 0 (not hazardous).

A.5 Logistic Regression

Logistic regression is a classification algorithm used to find the probability of event success and event failure. It applies to our data because our dependent variable is binary (safe/dangerous). Also, it is very fast at classifying unknown records.

	precision	recall	f1-score	support
0	1.00	0.99	1.00	186054
1	0.19	1.00	0.32	413
accuracy			0.99	186467
macro avg	0.60	1.00	0.66	186467
weighted avg	1.00	0.99	0.99	186467

Duration: 0:00:44.032736

Figure 15: *Metrics of the Logistic Regression in Validation Set*

A.6 Support Vector Machine (SVM)

SVM is used for classification, regression, and outliers detection. It is effective in high-dimensional spaces.

	precision	recall	f1-score	support
0	1.00	0.99	1.00	186054
1	0.25	1.00	0.40	413
accuracy			0.99	186467
macro avg	0.62	1.00	0.70	186467
weighted avg	1.00	0.99	1.00	186467

Duration: 0:25:00.098110

Figure 16: *Metrics of the Support Vector Machine (SVM) in Validation Set*

A.7 K-Nearest Neighbors (KNN)

The KNN algorithm classifies unclassified data points based on their proximity and similarity to other available data points. This algorithm's underlying assumption is that similar data points can be found near one another. It is very easy to understand and implement.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	186054
1	0.33	0.74	0.46	413
accuracy			1.00	186467
macro avg	0.67	0.87	0.73	186467
weighted avg	1.00	1.00	1.00	186467

Duration: 0:33:57.263505

Figure 17: *Metrics of the K-Nearest Neighbors (KNN) in Validation Set*

B. Neural Network

B.1 Multilayer Perceptron Model

MLP has a remarkable ability to derive meaning from complicated or imprecise data and can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. Two literature reviews mention that the MLP model was used in asteroid classification. So combining the structure of the MLP model in the two literature reviews, we used Keras to construct an MLP model with eight layers.

Model Structure:

Input Layer + Batch Normalization Layer

In the MLP model, we constructed one input layer containing ten neurons in the input layer and 45 neurons of input shape and used *relu* for our activation function. Besides, we also add one batch normalization handling the vanishing and exploding problem.

Hidden Layer + Batch Normalization Layer + Dropout Layer

Based on the suggestion in the literature review and the size of our large dataset, we chose to use 200 neurons in each hidden layer. The *relu* activation function followed two hidden layers, and we added one batch normalization layer after each hidden layer for handling the vanishing and exploding problem. We also add a dropout layer with a 0.05 dropout rate after the first batch normalization layer to prevent overfitting.

Output Layer

One output layer with one neuron, followed by the *sigmoid* activation function because we want the output to be defined as 0 (safe) or 1 (hazardous).

MLP Model

```
dl_model=Sequential()
dl_model.add(Dense(10, activation='relu', input_shape=(45,)))
dl_model.add(BatchNormalization())
dl_model.add(Dense(200, activation='relu'))
dl_model.add(BatchNormalization())
dl_model.add(Dropout(0.05))
dl_model.add(Dense(200, activation='relu'))
dl_model.add(BatchNormalization())
dl_model.add(Dense(1, activation='sigmoid'))
```

```
dl_model.summary()
```

```
Model: "sequential_39"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_156 (Dense)            (None, 10)                460
_________________________________________________________________
batch_normalization_117 (Bat (None, 10)                40
_________________________________________________________________
dense_157 (Dense)            (None, 200)               2200
_________________________________________________________________
batch_normalization_118 (Bat (None, 200)               800
_________________________________________________________________
dropout_39 (Dropout)         (None, 200)               0
_________________________________________________________________
dense_158 (Dense)            (None, 200)               40200
_________________________________________________________________
batch_normalization_119 (Bat (None, 200)               800
_________________________________________________________________
dense_159 (Dense)            (None, 1)                 201
=================================================================
Total params: 44,701
Trainable params: 43,881
Non-trainable params: 820
```

Figure 18: *MLP Model Summary*

Experimental Setup:

Performance

We used *binary_crossentropy* as our loss for our model compile since we are based on categorical data. Besides, we used accuracy as our compile metric.

Callback

1. *Model Checkpoint*: to save a model with the best weight, the best model with weight can be loaded later to continue the training from the state saved.
2. *Early Stopping*: Since we might face overfitting issues, we added early stopping with validation loss as our monitor, set patience as 15, and restored our best weights.
3. *Reduce LR On Plateau*: In our model, the learning rate of 0.001 decreases by a factor of 0.05 as soon as the validation loss has not improved for three consecutive steps.

Training Parameters

In our model, the learning rate greater than 0.001 caused the model to converge too quickly to a suboptimal solution. And, the learning rate of less than 0.001 is too low to drive the process to get stuck. Thus, we chose 0.001 as our learning rate. We chose 100 epochs and did not worry about spending too much time on training because we had early stopping. For the batch size, we had tried 32, 64, 96, and 128 and found that the model with batch sizes between 96 and 128 worked better. So we chose 110 as the batch size.

```
# Compile the model
dl_model.compile(optimizer=Adam(lr=0.001), loss='binary_crossentropy', metrics=['accuracy'])
```

```
# Set callback
model_checkpoint_cb=ModelCheckpoint(filepath='MLP_model.h5',
                                    save_best_only=True,
                                    save_weights_only=True)
early_stopping_cb=EarlyStopping(monitor='val_loss',
                                mode='min',
                                verbose=1,
                                patience=15,
                                restore_best_weights=True)
reducelr_on_plateau_cb=ReduceLROnPlateau(factor=0.05, patience=3)
```

```
start_time_dl=datetime.now()

dl_model_history=dl_model.fit(x_train_us_rus,y_train_us_rus,
                              epochs=100,
                              batch_size=110,
                              validation_data=(x_val,y_val),
                              callbacks=[model_checkpoint_cb, early_stopping_cb, reducelr_on_plateau_cb])
end_time_dl=datetime.now()
time_dl=str(end_time_dl-start_time_dl)
print('Duration:', time_dl)
```

Figure 19: *Experimental Setup in MLP Model*

Below is our training history and loss history of the MLP model.

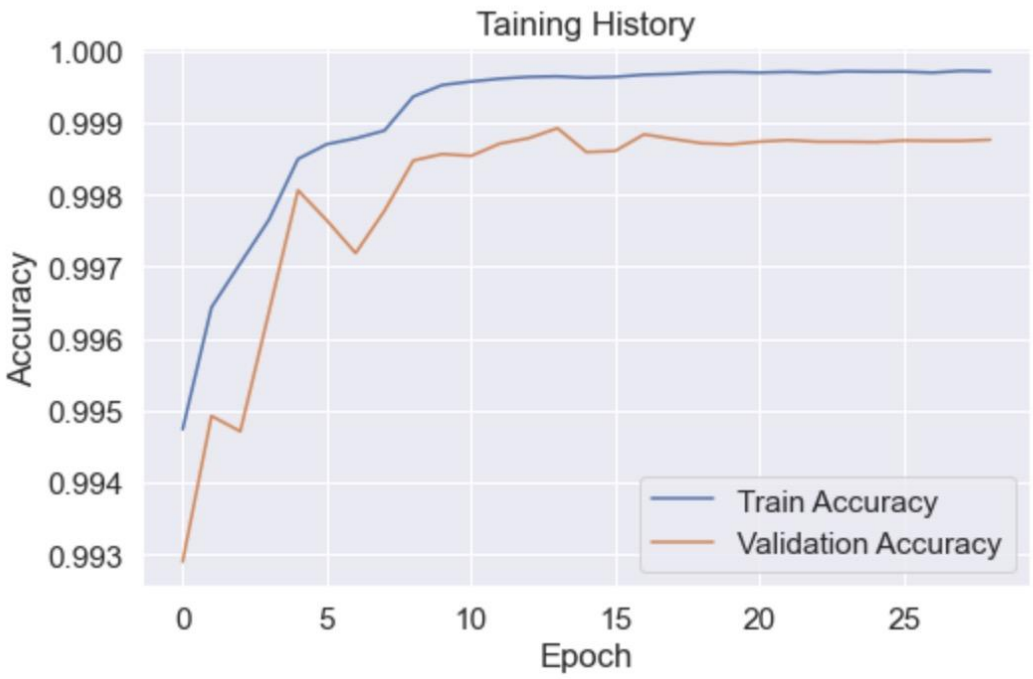

Figure 20: *Training History of MLP Model*

Figure 21: *Loss History of MLP Model*

	precision	recall	f1-score	support
0	1.00	1.00	1.00	186054
1	0.78	0.98	0.87	413
accuracy			1.00	186467
macro avg	0.89	0.99	0.94	186467
weighted avg	1.00	1.00	1.00	186467

Duration: 0:04:21.322164

Figure 22: *Metric of Multilayer Perceptron Model in Validation Set*

5. Results & Analysis

5.1 Model Comparison:

To facilitate the comparison of model performance, we created a model comparison table to compare precision, recall, F1 score, and running time among different models.

	ML Model	Precision	Recall	F1 Score	Time
1	Random Forest	0.831909	0.982508	0.893201	0:02:45.905846
2	XGBoost	0.831970	0.994601	0.897190	0:06:08.366335
3	Decision Tree	0.836387	0.982529	0.896341	0:00:07.427345
4	Navie Bayes	0.498893	0.500000	0.499446	0:00:00.985060
5	Logistic Regression	0.596181	0.995340	0.658987	0:00:44.032736
6	K-Nearest Neighbors	0.624997	0.995468	0.698235	0:25:00.098110
7	SVM	0.666739	0.866410	0.728819	0:33:57.263505
8	MLP Model	0.892420	0.990016	0.935793	0:04:21.322164

Figure 23: *Model Comparison Table*

According to the model comparison table, we chose MLP as our best model with the highest F1 score. Besides the MLP model, XGBoost performed best, and its F1 score was

0.897. The F-1 score of the decision tree is higher than the Random forest, one possible explanation is that the Decision Tree as a weak learner is overfitting our data, which performs a high f1 score on the validation set, but might result in low performance on the test set as it causes the high variance. RandomForest as an ensemble learner combines multiple weak learnings to avoid overfitting, which might cause lower performance on the training set but might get better performance on the test set. Naive Bayes performed worst of the eight models, with the lowest F1 score, because it works well when all features are independent. In our dataset, the features are correlative, so the Naive Bayes predicted that all planets would be safe. It is tough to obtain a complex relationship using logistic regression, so we set the logistic regression as our based model. Although SVM is effective in high dimensional space, it is more effective when the number of dimensions is greater than the number of samples. Also, SVM has the slowest computation speed. For KNN, if the chosen k value is incorrect, the model will be under or over-fitted to the data.

5.2 Best Model – MLP:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	186054
1	0.84	0.92	0.88	413
accuracy			1.00	186467
macro avg	0.92	0.96	0.94	186467
weighted avg	1.00	1.00	1.00	186467

Figure 24: *Metrics of Multilayer Perceptron Model in Test Set*

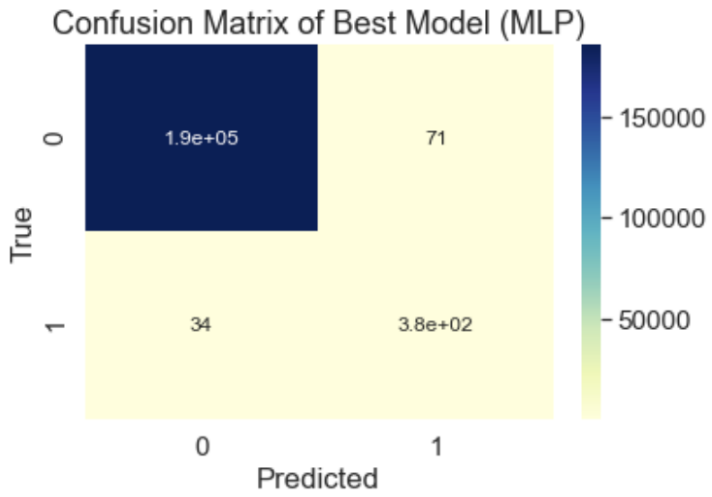

Figure 25: *Confusion Matrix of Multilayer Perceptron Model in Test Set*

The best model MLP got a 0.94 F1 score in the test set. From the above Figure 24, we could see our MLP model predicted all safe asteroids correctly. The precision of hazardous asteroids is 0.84 means from the detected hazardous asteroids, and 84% are actually hazardous. The recall of hazardous asteroids is 0.92 means that this MLP model correctly detected 92% of hazardous asteroids over total hazardous asteroids.

From the above Figure 25 confusion matrix, we could see that most data are classified correctly, and a small of them exited misclassified issues. Such as 34 potentially hazardous asteroids are misclassified as non-hazardous asteroids, and 71 non-hazardous asteroids are misclassified as potentially hazardous asteroids. The recall is a useful metric in the case of asteroid detection, where we want to minimize the number of False Negatives (34 in the confusion matrix) for any practical use since we don't want our model to detect a dangerous asteroid as safe. On the other hand, predicting a safe asteroid as dangerous (71 in confusion matrix) is not a big issue since in further exploration, it will be clear that it is not a threat to the Earth.

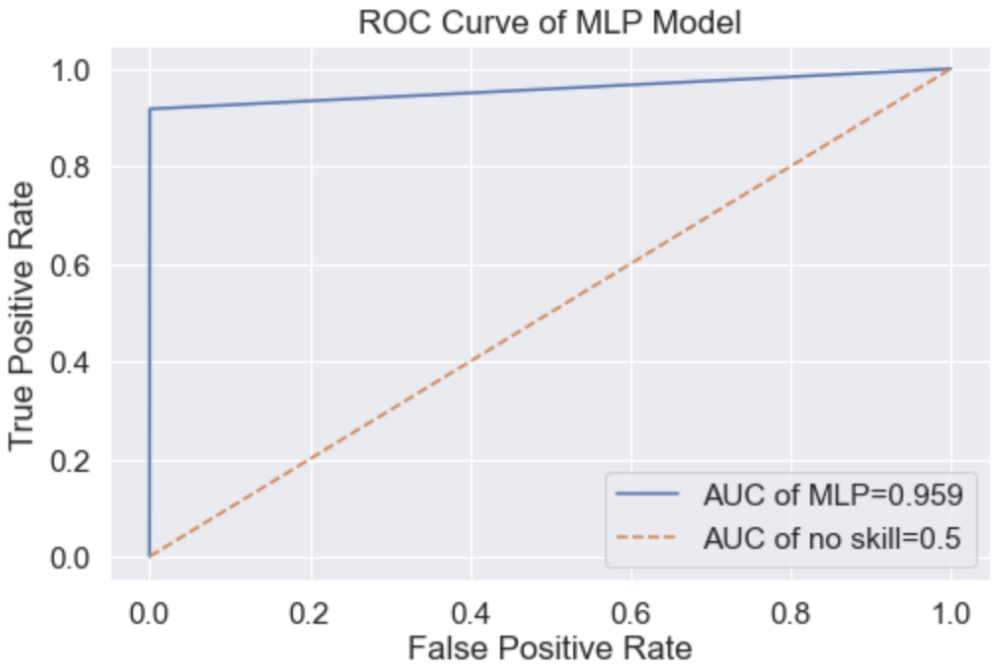

Figure 26: *ROC Curve of the MLP Model*

The more that the curve hugs the top left corner of the plot, the better the model does at classifying the data into categories. The AUC tells us how much of the plot is located under the curve. The closer AUC is to 1, the better the model. As we can see from the plot above, this MLP model does an excellent job of classifying the data into categories, and the AUC of this MLP model is 0.959.

6. Conclusion

6.1 Conclusion:

We designed, constructed, and trained a fairly simple neural network to classify asteroids with the potential to impact the Earth in the future. Our method takes the asteroid information (identify elements, size elements, and observed orbital elements) as input and provides a classifier for the expectation value for the object's striking Earth. Machine learning models can save thousands of person-hours and make automatic processing pipelines in astronomy much more efficient. We found that neural network models have slightly more success than linear and other simple models and have significantly more improvement capacity. However, neural network models tend to be more opaque and difficult to train. We can train neural networks without sacrificing too much explainability by using saliency maps such as the gradient explanation and keeping close track of the model's confusion matrix on the test set. To ensure the accuracy of our model in detecting potentially hazardous asteroids, it will be necessary in the future to clean up our data set and adjust our model to improve performance before deploying the best models on actual systems.

6.2 Project Limitation:

Our project has two limitations. The first one is cost constraint. Our data set was too large, but we didn't have enough project budget to support us running GPC on the virtual machine instances. Google Collaboratory's GCP did nothing to speed up our model. This makes it impossible to tune multiple options for multiple hyperparameters in a model at the same time because it would take nearly three hours to have three options for just one hyperparameter in a model. There is still room for optimization in our machine learning models, but cost constraints made them impossible to implement. The second limitation is data limitation. In our original data, there are three columns of missing values about the size characteristics of the planet with more than 86%. To not affect the model's effectiveness, we deleted these three columns. The more the data on the characteristics of asteroids, the more likely we are to get the best models. So if we had more features, our model would be better.

6.3 Future Research:

For future research, we will continue to pay attention to the data that are collected and updated by radar and infrared rays for the characteristics of asteroids because the orbit of asteroids will be changed by gravity, which may change the previously non-potentially hazardous asteroids to PHA, and vice versa. Therefore, we should combine the other data sources about the different features and continuously update the model according to the latest data to improve the prediction rate of dangerous asteroids. In other words, it increases the recall rate while maintaining the high precision of predicting hazardous

asteroids, which means minimizing the number of False Negatives and minimizing the number of asteroids that are actually hazardous but are predicted to be safe.

Completely Capstone Project in Rayna Liu Github and Zheyue Wang Github.
Completely python code in Github Page.

7. References

[1] NASA. (2022, January 4). *Asteroids*. NASA. Retrieved February 23, 2022, from https://solarsystem.nasa.gov/asteroids-comets-and-meteors/asteroids/overview/?page=0&per_page=40&order=name%2Basc&search=&condition_1=101%3Aparent_id&condition_2=asteroid%3Abody_type%3Ailike

[2] McFall-Johnsen, M. (2021, May 12). *A NASA simulation revealed that 6 months' warning isn't enough to stop an asteroid from hitting Earth. we'd need 5 to 10 years.* Business Insider. Retrieved February 23, 2022, from https://www.businessinsider.com/nasa-asteroid-simulation-reveals-need-years-of-warning-2021-5

[3] Kumar, S. (2020, May 1). *Data splitting technique to fit any machine learning model*. Medium. Retrieved March 15, 2022, from https://towardsdatascience.com/data-splitting-technique-to-fit-any-machine-learning-model-c0d7f3f1c790

[4] NASA Asteroid Classification https://towardsdatascience.com/nasa-asteroid-classification-6949bda3b1da

[5] Prediction of Orbital Parameters for Undiscovered Potentially Hazardous Asteroids Using Machine Learning https://trello.com/c/9mWT7AdD/74-prediction-of-orbital-parameters-for-undiscovered-potentially-hazardous-asteroids-using-machine-learning

[6] Kumar, A. (2020, September 4). *Micro-average & Macro-average Scoring Metrics - Python*. Data Analytics. https://vitalflux.com/micro-average-macro-average-scoring-metrics-multi-class-classification-python/\

8. Appendix

8.1 Correlation Matrix

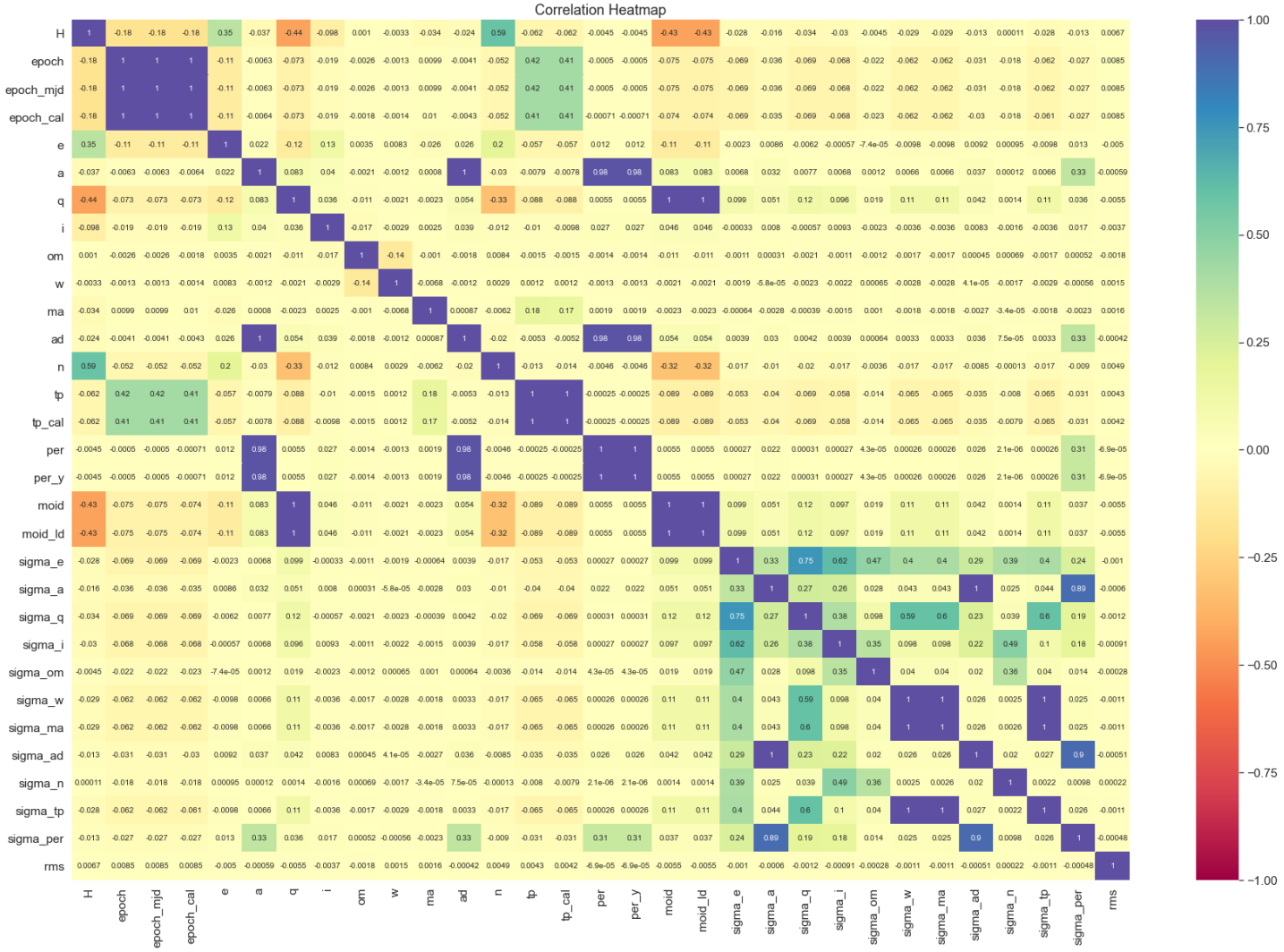

Figure 27: *Correlation Matrix of all numerical features*

The correlation matrix above shows the correlations between all numerical features before encoding categorical features. *epoch* has a strong relationship with *epoch_mjd* and *epoch_cal*, they both refer to epochs of osculation in different time units. *tp* has a strong relationship with *tp_cal*, they refer to the time of perihelion passage in different time units. *per* has a strong relationship with *per_y*, they refer to sidereal orbital periods in a different time unit. *moid* has a strong relationship with *moid_ld*, they refer to earth minimum orbit intersection distance in different length units. *a* has a strong relationship with *ad* and *per*. *q* has a strong relationship with *moid*. *sigma_w* has a strong relationship with *sigma_ma* and *sigma_tp*.

8.2 Decision Tree Plot

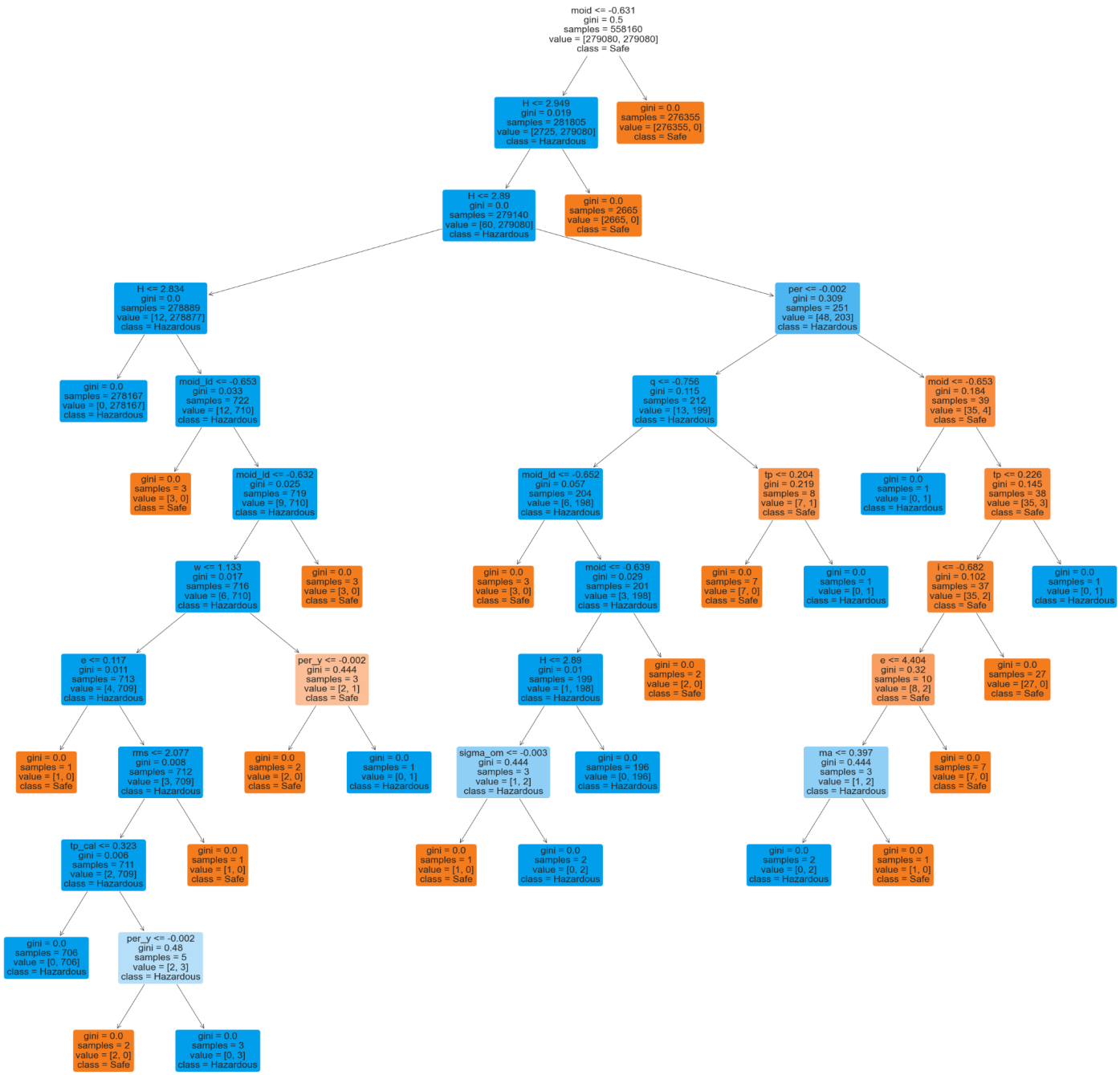

Figure 28: *Plot of Decision Tree*

The above graph shows that the decision tree is a classifier based on the different thresholds of *moid* and *H*. Asteroids are identified as potentially hazardous asteroids if their *moid* is greater than -0.631 and *H* is equal and less than 2.949.