Asteroid Classification Data

Ankita Giri, Sihua Cai, Jaehyeon Park, Yuxi Jiang

May 8, 2022

Abstract

The Kaggle Asteroid Classification Data has been used for predictive modeling (mostly supervised) widely for predicting hazardous nature of asteroids which is a binary value. The dataset is affected by multicollinearity, outliers, redundancy of features, inconsistent features, and a disbalanced proportion of classification data points. To address the issues aforementioned, we apply data preprocessing methods, use information extraction processes appropriate to the nature of the data set which narrows down to 16 features from 40 features. We use Variance Inflating Factor (VIF) values to identify multicollinearity, and then for information extraction process we use exploration methods like building categorically faceted density plots, building contingency table by binning of continuous data into categorical intervals, apply l2-regularized shrinkage method ridge regression which is robust to multicollinearity. We do a final check for the predictive capability of the processed data by applying appropriate statistical learning models. Our main goal is to identify what makes an asteroid hazardous.

1 Introduction

1.1 Dataset

In this project, we will work with a Kaggle dataset that contains Asteroid data from the NASA API. The "NASA: Asteroids Classification" dataset provides data on measurements and characteristics for 4,687 asteroids. For each asteroid observation, 40 such variables are provided, including a binary indicator variable that specifies whether an asteroid is hazardous or not. There are 40 column variables of the original dataset.

The variables in the dataset are as follows:

1. `Neo Reference ID`
2. `Name`
3. `Absolute Magnitude`
4. `Estimated Diameter in KM (min)`
5. `Estimated Diameter in KM (max)`
6. `Estimated Diameter in M (min)`
7. `Estimated Diameter in M (max)`
8. `Estimated Diameter in Miles (min)`
9. `Estimated Diameter in Miles (max)`
10. `Estimated Diameter in Feet (min)`
11. `Estimated Diameter in Feet (max)`
12. `Close Approach Date`
13. `Epoch Date Close Approach`
14. `Relative Velocity KM per sec`
15. `Relative Velocity KM per hr`
16. `Relative Velocity Miles per hr`
17. `Miss Distance (Astronomical)`
18. `Miss Distance (Lunar)`
19. `Miss Distance (KM)`
20. `Miss Distance (Miles)`
21. `Orbiting Body`
22. `Orbit ID`
23. `Orbit Determination Date`
24. `Minimum Orbit Intersection`
25. `Orbit Uncertainty`
26. `Jupiter Tisserand Invariant`
27. `Epoch Osculation`
28. `Eccentricity`
29. `Semi Major Axis`
30. `Inclination`
31. `Ascending Node Longitude`
32. `Orbital Period`
33. `Perihelion Distance`
34. `Perihelion Argument`
35. `Aphelion Distance`
36. `Perihelion Time`
37. `Mean Anomaly`
38. `Mean Motion`
39. `Equinox`
40. `Hazardous`

These variables describe various features of an asteroid, including but not limited to

1. Physical measurements of the asteroid body and its motion, such as Absolute Magnitude, Maximum and Minimum Estimated Diameter, and Relative Velocity. We are also clued into some of the geometrical features of the asteroid through information from variables such as Semi Major Axis and Ascending Node Longitude.
2. Orbital patterns and paths. This is illustrated through variables such as Eccentricity and Orbital Period. We can also learn more about the

asteroid's orbit relative to Earth and the Sun by looking at variables such as Miss Distance, Perihelion Distance, and Aphelion Distance.

3. Binary indication of hazardous or non-hazardous state.

1.2 Brief Explanation of Project

In this project, we hope to achieve two goals. First, and most importantly, we aim to identify potential hazardous and non-hazardous asteroid bodies from the data given, applying methods such as ridge regression and other statistical machine learning methods. Secondly, through trying various methods, we hope to pinpoint the method that is optimized for our goal of predicting potential hazardous asteroids.

We hypothesize that the following feature may have a large influence on whether an asteroid is classified as hazardous or non-hazardous:

1. Miss Distance
   Miss distance is defined as the distance of an asteroid to Earth at the point in the asteroid's orbit where it is closest to Earth. Effectively, it can be considered the distance by which the asteroid "missed" coming into contact with Earth. Asteroids with a lower miss distance are close to a potential collision with Earth's surface, and thus we believe that Miss Distance may be an important feature to note.

2. Orbit Uncertainty
   An asteroid with increased values of orbit uncertainty poses a potential threat to collision, since the trajectory cannot be predicted as well.

Of course, as students with a limited background in astronomy, these are simply our preliminary speculations. We are excited to see if the above mentioned features have as much weight as we think they do.

2 Methodology

2.1 Data Cleaning

Dropping Redundant Variables

Measurement Units:

The `Est Dia` variables, which indicate the asteroid's minimum and maximum estimated diameter measurements, and asteroid velocity variables appear in the dataset many times under different units. For easier and more accurate interpretability, a consistent system of measurement in the form of SI units was used. Thus, the a new variable with measurement of distance as meters(m) and measurement of time as seconds(s) is created.

Following that, the default diameter and velocity variables `Est Dia in KM(min)`, `Est Dia in KM(max)`, `Est Dia in M(min)`, `Est Dia in M(max)`, `Est Dia in Miles(min)`, `Est Dia in Miles(max)`, `Est Dia in Feet(min)`, `Est Dia in Feet(max)`, `Miss Dist.(Astronomical)`, `Miss Dist.(Lunar)`, `Miss Dist.(Miles)`, `Relative Velocity km per sec`, and `Miles per hour` are removed from our working dataset.

Other Redundant Variables:
Additionally, the variables `Orbiting Body` and `Equinox` only had one unique value ("Earth" and "J2000", respectively). Identification variables `Name`, `Orbit ID`, and `Neo Reference ID` would not generate insight to whether an asteroid is hazardous or not. A similar result is true of the three date variables, `Close Approach Date`, `Epoch Date Close Approach`, and `Orbit Determination Date`. Including these variables in our analysis would not generate new information, so the variables are dropped.

Imbalanced Dataset

As the histogram showed, the dataset about `'hazardous'` and `'non-hazardous'` is considerably imbalanced. The amount of the `'hazardous'` data is 755, and the `'non-hazardous'` data is 3932. In other words, around 83.89% of the dataset is `'non-hazardous,'` and only 16.11% is `'hazardous.'` It means that even if we make a model which predicts all the variables as `'not hazardous,'` we could get 83.89% accuracy. It is not easy to depend on accuracy to evaluate a machine learning classifier trained on this dataset. Therefore, resampling the dataset is needed, and oversampling is one way to edit this dataset.

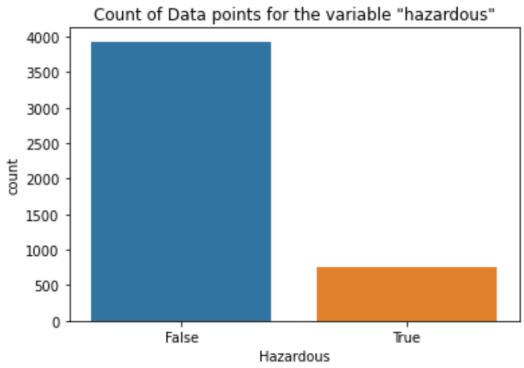

Figure 1: Barplot showing the imbalanced amount of hazardous and non-hazardous data. True indicates hazardous asteroid and False indicates non-hazardous asteroid

Standardization of Data

The variables are all measurements of different quantities and have different

measurement of units. Some variables like Relative Velocity (m per s) have values up to 40,000 while the maximum value of Minimum Orbit Intersection is only 0.4. Our project uses model like Ridge Regression which requires the data to be at a similar scale. Therefore, the data has been standardized to be at a scale of E(X) = 0 and Var(X) = 1 when applying certain models.

Outliers

Detecting and removing outliers is important because outliers affect data analysis and total distribution. It also affects estimation and prediction. Since the univariate method correlated with one variable, we used the boxplots to detect the outliers. As we can see from the boxplots below, most outliers are global and collective outliers. In addition, we used the `'IQR'`(Inter Quartile Range) method to detect and remove the outliers. We set `'IQR'` as (75th percentile of the data) - (25th percentile of the data), and (25th percentile of the data)–1.5IQR represent the smallest value in the dataset, and (75th percentile of the data+1.5IQR) represent the largest value in the dataset. Otherwise, we treated the values as outliers and removed them.

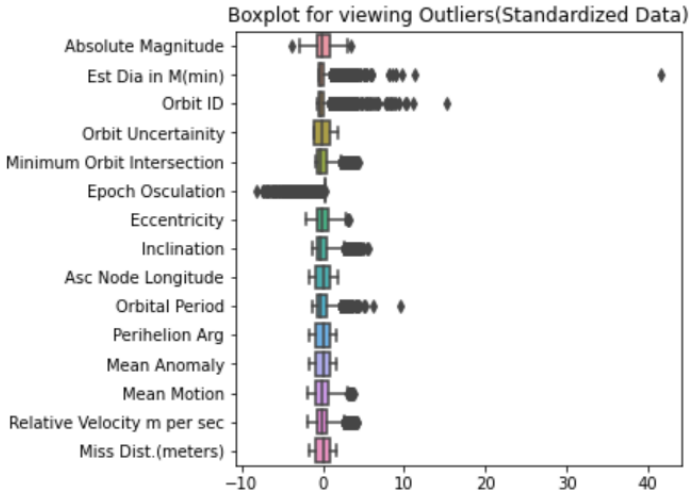

Figure 2: Boxplot of the Outliers of the Features

2.2 Handling of Multicollinearity

Why care about multicollinearity?

One of the basic assumptions of a regression model is that the X matrix is a full column rank matrix(predictors assumed to be linearly independent). This is not held true in a design matrix with multicollinearity. The condition of the matrix X, $\kappa(X) = X^T X$ is high. Since $\hat{\beta_n} = (X^T X)^{-1}.X^T Y$, the value of $\hat{\beta_n}$

decreases when $X^T X$ increases. Also, $\hat{\beta}_n = \arg\min \text{RSS}(\beta)$, so the decrease in value of $\hat{\beta}_n$ implies a decrease in the residual sum of squares(RSS). This causes an overfit of the model, decreasing accuracy and interpretability of the results. Most supervised modeling methods appropriate for the data used in this project are a form of OLS, we take extra steps to detect the multicollinearity(using Variance Inflation Factor) and use methods that are robust to multicollinearity to further generate information from the data.

Pairwise Correlation Matrix:

Before we look at the VIF values, we view a pairwise correlation matrix as a diagnostic measure. The correlation matrix below, shows that there are multiple variables that are correlated to each other. Some correlated variables are:

1. Perihelion Time and Epoch Osculation

2. Aphelion Dist and Semi Major Axis, Orbital Period, Aphelion Dist, Mean Motion, Jupiter Tisserand Invariant, Perihelion Distance3. Absolute Magnitude, Orbit ID, Orbit Uncertainty, Est Dia in M(min), Est Dia in M(min)

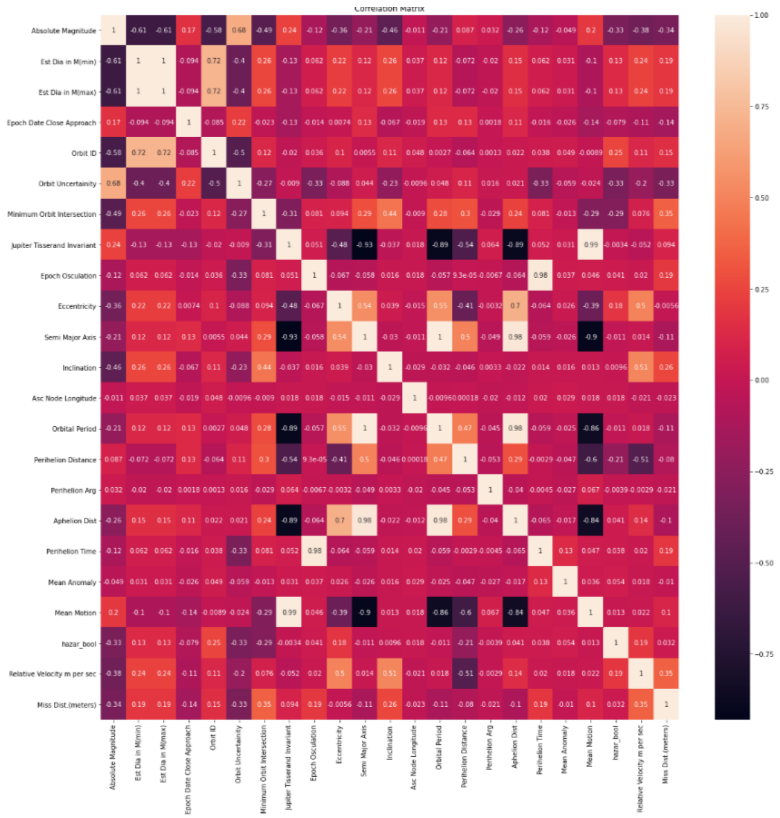

Figure 3: Correlation Matrix of the Variables

Variance Inflation Factor(VIF):

Multicollinearity occurs if multiple variables are correlated with each other. Looking at pairwise correlation values does not show whether there is a presence of collinearity between multiple variables. One way to detect multicollinearity is checking for the VIF values. VIF is calculated using R^2 values calculated from Multiple Linear Regression by keeping one variable as the response and regressing the rest of the variables as the independent variable.

$$VIF_i = \frac{1}{1 - {R^2}_i} \tag{1}$$

Predictors with large VIF values indicate that the variable of the weight coefficient is inflated by a factor of the calculated VIF value because of the presence of high correlation between at least one other variable.[1]

In this case, predictors with the largest VIF values are dropped one at a time until all the VIF values are less than or equal to 5. Below the tables show VIF values at different iteration stages which are sorted in descending order according to the VIF value. The variables with the largest VIF value are not dropped all at once but one at a time. Because the variables were dropped one at a time, it can be seen that the order index of the variables have changed at each iteration. For instance, going from 6th to 7th iteration, the variable Eccentricity's sorted index dropped from 2 to 4. If all the variables with high VIF were dropped at once, it would have dropped all the possibly important variables too.

Although the large VIF values have been dropped and the threshold for the largest VIF value was kept an arbitrary value of less than or equal to 5, it still indicates that there could be some multicollinearity. Thus, a need for further visualization and information generating techniques that is robust to multicollinearity is necessary to extract interpretable results. Hence, we use shrinkage method like ridge regression to look at the coefficents.

Note: As VIF is calculated using the R^2 values calculated from Multiple Regression, the imbalanced nature of the data might have affected some results. To compensate for that, a comprehensive check for variables was done when looking at the density plots(mentioned below).

	Variable	VIF
3	Est Dia in M(max)	inf
2	Est Dia in M(min)	inf
10	Semi Major Axis	9.007199e+15
16	Aphelion Dist	9.285772e+13
14	Perihelion Distance	1.491505e+12
7	Jupiter Tisserand Invariant	1.282576e+03
13	Orbital Period	1.039371e+03
19	Mean Motion	7.964119e+02
17	Perihelion Time	2.769625e+01
8	Epoch Osculation	2.751655e+01
9	Eccentricity	1.903914e+01
11	Inclination	5.316031e+00
1	Absolute Magnitude	4.834677e+00
21	Relative Velocity m per sec	3.411821e+00
6	Minimum Orbit Intersection	2.772650e+00
4	Orbit ID	2.762411e+00
5	Orbit Uncertainity	2.520312e+00
20	hazar_bool	1.666872e+00
22	Miss Dist.(meters)	1.622994e+00
18	Mean Anomaly	1.238061e+00
15	Perihelion Arg	1.012115e+00
12	Asc Node Longitude	1.008111e+00

	Variable	VIF
6	Jupiter Tisserand Invariant	720.324909
16	Mean Motion	578.205741
14	Perihelion Time	27.690919
7	Epoch Osculation	27.509866
11	Orbital Period	18.262216
12	Perihelion Distance	16.955384
8	Eccentricity	15.155184
1	Absolute Magnitude	4.779744
9	Inclination	3.772857
18	Relative Velocity m per sec	3.396089
5	Minimum Orbit Intersection	2.754807
3	Orbit ID	2.754623
2	Est Dia in M(min)	2.631702
4	Orbit Uncertainity	2.513684
17	hazar_bool	1.663840
19	Miss Dist.(meters)	1.621270
15	Mean Anomaly	1.237951
13	Perihelion Arg	1.011063
10	Asc Node Longitude	1.008000

	Variable	VIF
11	Perihelion Distance	16.624631
7	Eccentricity	13.619609
10	Orbital Period	8.802389
14	Mean Motion	8.065224
1	Absolute Magnitude	4.744188
16	Relative Velocity m per sec	3.197051
5	Minimum Orbit Intersection	2.754166
3	Orbit ID	2.747692
2	Est Dia in M(min)	2.603923
4	Orbit Uncertainity	2.504953
8	Inclination	2.380622
15	hazar_bool	1.661511
17	Miss Dist.(meters)	1.592252
6	Epoch Osculation	1.200411
13	Mean Anomaly	1.012483
12	Perihelion Arg	1.009681
9	Asc Node Longitude	1.007622

	Variable	VIF
10	Orbital Period	5.186616
1	Absolute Magnitude	4.744143
13	Mean Motion	4.057247
7	Eccentricity	2.935059
15	Relative Velocity m per sec	2.882509
3	Orbit ID	2.722022
2	Est Dia in M(min)	2.589699
5	Minimum Orbit Intersection	2.587552
4	Orbit Uncertainity	2.490373
8	Inclination	2.359751
14	hazar_bool	1.659931
16	Miss Dist.(meters)	1.573788
6	Epoch Osculation	1.199449
12	Mean Anomaly	1.011573
11	Perihelion Arg	1.009681
9	Asc Node Longitude	1.007622

Table 1: VIF values showing dropping of variables with highest VIF at different iterations: Upper left- 1st iteration, Upper Right- 4th Iteration, Lower left- 6th iteration, Lower Right- 7th iteration. VIF values are sorted in descending order at each iteration.

2.3 Visualization and Information Extraction

Ridge Regression:

Ridge regression adds a penalty term to the loss function that is the squared magnitude value of the coefficient (l2- regularization).

$$RSS + \lambda \sum_{j=1}^{p} {\beta_j}^2 \tag{2}$$

where λ is the tuning parameter. The penalty term $\lambda \sum_{j=1}^{p} {\beta_j}^2$ is a shrinking coefficient that will penalize coefficients to shrink to 0 as the $\lambda \to \infty$. Ridge regression was motivated as a fix to extracting well defined estimators of collinear covariates. The coefficient estimates of strongly positively correlated covariates shrink lesser when applying ridge regression model hence looking at ridge trace provides some insight to which variables maybe affecting the response. [1]

Looking at variables using ridge trace and coefficients shrinkage plot:

We use the data that has been standardized to achieve equal variance for ridge regression. Also, the data has been adjusted to be more balanced before application of the model. Explanation of the imbalanced nature of the data and its adjustment is mentioned in the Data Cleaning part and also Oversampling(2.4).

Below the figure (left)is a ridge trace plot with the optimal λ value(dotted line)selected by performing 10-fold Cross validation. As it can be seen in the figure(right) which is a plot of coefficients at the optimal lambda value selected, variables like Minimum Orbit Intersection, Absolute Magnitude, Est Dia in M(min), Orbit Certainty shrink the slowest. We further look deeper into the variables to extract more information.

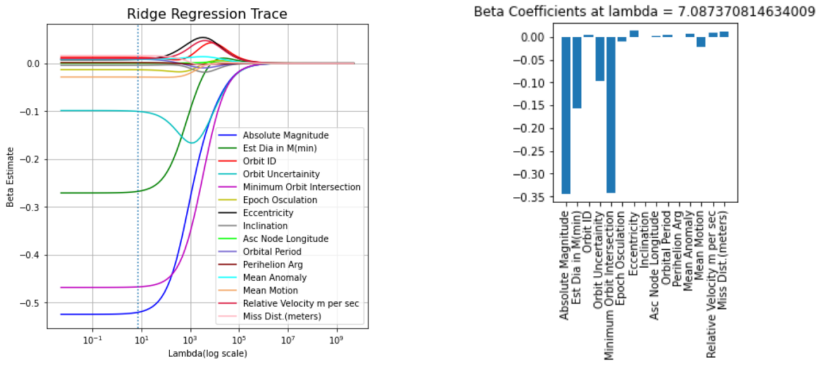

Figure 4: (Left) Ridge Trace for the variables(Standardized)- dotted line is the optimal lambda value selected by 10-fold cross validation, (Right) Coefficients estimates at the optimal lambda value

Note: Ridge regression is a bias- variance tradeoff model which means that to reduce the variance, more bias is introduced in the estimators. This is also a reason why the MSE value first decreases in ridge and sharply increases as the lambda value increases. A high bias causes the algorithm to potentially miss relevant features thus, ridge regression is not the best method for feature selection. However, the coefficient estimates that shrink slower may carry useful information regarding the response so we use the ridge trace to look at the coefficient estimates that shrink slower more closely.

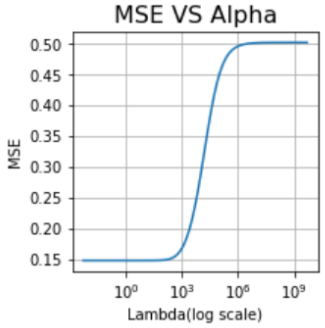

Figure 5: MSE as the λ value increases

Density Plots:

With the data being imbalanced, visualization of distribution of the data with density plots provided a clearer picture about the patterns in the data. The plots are faceted with by the categorical hazardous variable where 0 indicates that the asteroids are non-hazardous and 1 indicates that the variables are hazardous.

From the ridge trace plots, some patterns to explain hazardous behavior in variables like Absolute Magnitude, Orbit Uncertainty, Minimum Orbit Intersection, Est Dia in M(min) was expected to be seen. The pattern among these variables as shown in figure below.

These plots show noticeable difference between distribution of the the hazardous and non-hazardous were seen in the variables. For instance, the Absolute Magnitude variable when faceted by the variable hazardous nature of the variable, showed that there is a significant spike in the hazardous variables between ranges 20-25. Another noticeable pattern was for Minimum Orbit Intersection for which non- hazardous distribution has a spike in the range 0-0.1. Likewise, there are visible patterns in the distribution for Orbit Uncertainty.

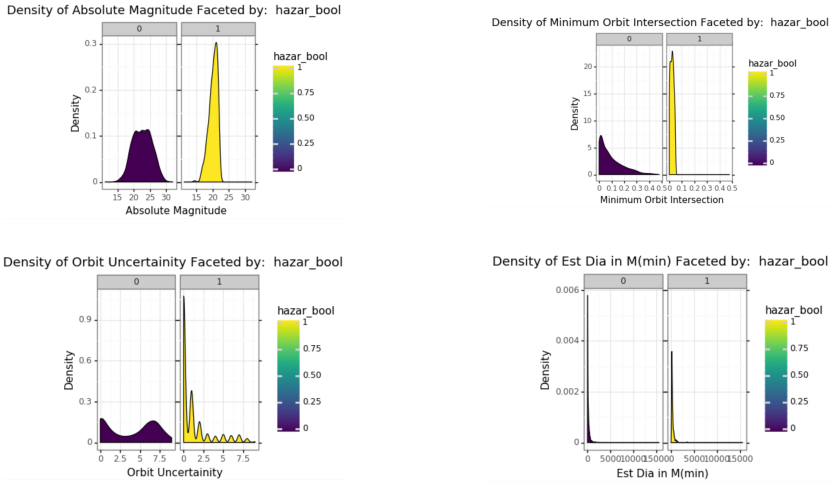

Figure 6: Density of some variables as a function of hazardous nature of the asteroid. 0 indicates non-hazardous, 1 indicates hazardous

Some other variables that have a slight but not a explainable difference in distribution were Perihelion Arg, Eccentricity shown below. Thus we further use contingency tables for deeper understanding of these variables.

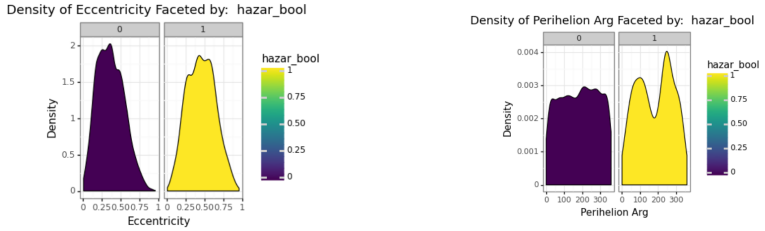

Figure 7: Density Plots for some additional variables

Contingency Tables:

Contingency tables displays the frequency of distribution of variables which makes it a great way to view patterns within a variable. We divide the continuous variables to bins (intervals) as categorical features to see patterns. As the data is imbalanced, the proportion of the categorical features is used for creating the contingency tables.

Figure below shows contingency tables and noticeable patterns that are seen in the variables.

Min Orbit Intersection	(0.0, 0.025]	(0.025, 0.05]	(0.05, 0.1]	(0.1, 0.2]	(0.2, 0.3]	(0.3, 0.4]
hazar_bool						
0	0.317248	0.107290	0.216632	0.220226	0.10806	0.030544
1	0.561589	0.438411	0.000000	0.000000	0.00000	0.000000

All the hazardous asteroids in the data have Minimum Orbit Intersection value below 0.5. The largest proportion of hazardous fall in the range between the values 0.01-0.05.

Est Diameter(m)	(0, 200]	(200, 400]	(400, 600]	(600, 800]	(800, 1000]	(1000, 1200]	(1200, 1400]	(1400, 16000]
hazar_bool								
0	0.725331	0.153103	0.062818	0.023906	0.013225	0.006612	0.004069	0.010936
1	0.413245	0.361589	0.120530	0.047682	0.022517	0.018543	0.010596	0.005298

The larger the Estimated Diameter, the larger the proportion of hazardous asteroids than non-hazardous asteroids.

Abs Magnitude	(10, 15]	(15, 18]	(18, 19]	(19, 20]	(20, 21]	(21, 22]	(22, 23]	(23, 25]	(25, 30]	(30, 35]
hazar_bool										
0	0.001526	0.044252	0.068159	0.093337	0.118260	0.098678	0.116989	0.228128	0.2294	0.001272
1	0.002649	0.080795	0.117881	0.214570	0.286093	0.282119	0.015894	0.000000	0.0000	0.000000

All the asteroids that are hazardous has Absolute Magnitude value below 23.

Orb Uncertainty	(0.5, 1.0]	(1.0, 2.0]	(2.0, 4.0]	(4.0, 6.0]	(6.0, 8.0]	(8.0, 10.0]
hazar_bool						
0	0.139194	0.072927	0.116883	0.299034	0.331668	0.040293
1	0.453172	0.181269	0.129909	0.129909	0.099698	0.006042

A large proportion of hazardous asteroid have Orbit Certainty <0.5. A noticeable pattern that is seen- lower the orbit uncertainty, the larger the proportion of the asteroid are hazardous.

Eccentrcity	(0.0, 0.2]	(0.2, 0.4]	(0.4, 0.6]	(0.6, 0.8]	(0.8, 1.0]
hazar_bool					
0	0.198627	0.390895	0.305188	0.096897	0.008393
1	0.082119	0.298013	0.381457	0.203974	0.034437

Larger proportion of asteroids with eccentricity value >0.4 are hazardous.

Perihelion Dist	(0.0, 0.2]	(0.2, 0.4]	(0.4, 0.6]	(0.6, 0.8]	(0.8, 1.0]	(1.0, 1.2]
hazar_bool						
0	0.003763	0.038978	0.154839	0.249462	0.310484	0.242473
1	0.015894	0.074172	0.259603	0.268874	0.332450	0.049007

It can be seen that lower Perihelion distance may be hazardous.

Concluding remarks from the contingency table are put in a table below. It is important to note that the binning of values into continuous intervals causes a loss of information and the interpretation does not provide a tangible explanation but provides a rather closer approximated explanation of the variables.

Feature	Notable Pattern
Minimum Orbit Intersection	Value <0.5 are hazardous; All hazardous between range 0.01-0.05
Estimated Diameter(m)	Larger proportion of hazardous for larger estimated diameter
Absolute Magnitude	All hazardous range <23
Orbit Uncertainty	Large proportion of hazardous in range <0.5. Lower the value, larger the proportion of hazardous
Eccentricity	large proportion of hazardous in range >0.4
Perihelion Distance	lower perihelion distance may be hazardous

Table 2: Summary of information regarding hazardous nature of asteroid from Contingency Tables

2.4 Prediction methods

Oversampling

In this project, we chose to use `oversampling`, which duplicates examples from the minority class (i.e. 'hazardous' in this dataset) in the training dataset before fitting a model to fix the problem that there are too few examples of minority class to learn. As this dataset is imbalanced, we need to resample the dataset. However, if we duplicate the same value from the minority data population, there will be a high chance of overfitting. Therefore, we chose `SMOTE` (Synthetic Minority Oversampling Technique) for oversampling and randomly sampled data from the minority class, then k-nearest neighbors from the data are set. Synthetic data would then be made between the random data and the randomly selected k-nearest neighbors. This process will be repeated many times until the minority class has the same proportion as the majority class ('non-hazardous). It significantly reduced the chance of overfitting as compared to classic oversampling.[2]

It is important to note that using `SMOTE` method for oversampling has some

drawbacks. It oversamples values that does not carry meaningful information. For modeling like Knn, oversampling makes it difficult to determine the number of nearest neighbors and is computationally expensive too as shown in the comparison table for all the algorithms below. [3]

Logistic Regression

Logistic regression model is a widely used algorithm for classification problems when there is a binary response variable. There are six main assumptions of the model. 1) Response variable is binary. 2) Observations are independent. 3) No multicollinearity among explanatory variables. 4) No significant outliers 5) Explanatory variables are linearly correlated with the logit of the response variable 6) The sample size is sufficiently large.[4] In our dataset, the labeling of asteroids as being hazardous or not is the target variable, and all other features/measurements of the asteroid are the explanatory variables. Before applying the model, we used methods like Ridge regression and Variance Inflation Factor to get rid of multicollinearity, and eliminate outliers that are beyond the 0.25/0.75 quantile. Lastly, since our dataset is relatively large(4687 entries), the prior assumptions of the Logistic regression model are all satisfied.

In our self-implemented model, we basically used the weight and bias term to calculate the weighted sum (i.e. $X * w + b$) and pass it through a `sigmoid`function($\frac{1}{1+e^{-x}}$) to get the estimated probability of a given asteroid being classified as hazardous. The value is between 0 and 1. As a result, any observation with value greater than 0.5 will be classified as 1(hazardous) or 0(non-hazardous). 0.5 is thus called the decision boundary.

To optimize the iteration process, we used the *Gradient Descent algorithm* to find the best values for weight/bias that minimize the loss function. To achieve this goal, we computed the partial derivative of loss function *w.r.t.* weight and bias. Then, according to the update rule

$$\begin{aligned} w &= w - \text{learning rate} * dw, \\ b &= b - \text{learning rate} * dw \end{aligned} \tag{3}$$

we kept updating the parameters until the optimized value was reached. Then, we applied the model on data before and after oversampling as a comparison. The run time for data is 1.3 and 2.02 seconds, respectively, with and without oversampling. Classification reports are shown below.

	0	1	accuracy	macro avg	weighted avg
precision	0.839590	0.0	0.83959	0.419795	0.704912
recall	1.000000	0.0	0.83959	0.500000	0.839590
f1-score	0.912801	0.0	0.83959	0.456401	0.766379
support	984.000000	188.0	0.83959	1172.000000	1172.000000

Table 3: Classification Report(Without oversampling)

	0	1	accuracy	macro avg	weighted avg
precision	0.798122	0.716168	0.751696	0.757145	0.757701
recall	0.682731	0.822558	0.751696	0.752645	0.751696
f1-score	0.735931	0.765685	0.751696	0.750808	0.750606
support	747.000000	727.000000	0.751696	1474.000000	1474.000000

Table 4: Classification Report(With oversampling)

Naïve Bayes

Naïve Bayes algorithm is based on *Bayes' theorem* and assumes independence among predictor variables.[5] *Bayes' theorem* can be expressed as the calculation of conditional probability:

$$p(C_k|x_i) = \frac{p(x_i)p(x_i|C_k)}{p(x_i)}, \tag{4}$$

where C_k indicates k different classes, $p(x_i)$ is the prior probability and $\frac{p(x_i)p(x_i|C_k)}{p(x_i)}$ is the joint probability.

In our self-implemented model, we first calculated the prior and posterior probabilities using training and test data. Note that we calculated posterior probabilities using Multivariate normal distribution formula, Then, in order to find the class with the highest conditional probability $p(C_k|x_i)$, we set

$$y = argmax_y \frac{p(x_i)p(x_i|C_k)}{p(x_i)}. \tag{5}$$

To simplify the calculation, we only took the log of the numerator and got

$$y = argmax_y \sum_{i=1}^{n} log(p(x_i)). \tag{6}$$

This is the formula that we used to find the most probable class. [6]

Then, we applied the model on data before and after oversampling as a comparison. The run time for data is 0.21 and 0.12 seconds, respectively, with and without oversampling. Classification reports are shown below.

	0	1	accuracy	macro avg	weighted avg
precision	0.839590	0.0	0.83959	0.419795	0.704912
recall	1.000000	0.0	0.83959	0.500000	0.839590
f1-score	0.912801	0.0	0.83959	0.456401	0.766379
support	984.000000	188.0	0.83959	1172.000000	1172.000000

Table 5: Classification Report(Without oversampling)

	0	1	accuracy	macro avg	weighted avg
precision	0.950954	0.933784	0.942334	0.942369	0.942485
recall	0.934404	0.950481	0.942334	0.942443	0.942334
f1-score	0.942606	0.942059	0.942334	0.942332	0.942336
support	747.000000	727.000000	0.942334	1474.000000	1474.000000

Table 6: Classification Report(With oversampling)

Support Vector Machine

Support Vector Machine is a supervised learning technique that is commonly used in both regression analysis and classification problems. It's flexible and can be used to classify linear and non-linear data. There are two main assumptions under this model: 1) All observations are independently and identically distributed. 2) The margin is at its biggest.[7] The main idea is to separate data into different classes/labels and use a hyperplane to maximize the margin while minimizing the loss.

In our self-implemented model, we first set up a hyperplane($w * x - b = 0$) to separate the asteroids into different classes: 1(hazardous) and -1(nonhazardous). We used the *hinge loss* function to find the maximum margin. When the observation is correctly classified, then there will be a 0 loss. Otherwise, loss will be equal to $1 - y_i * (w * x - b)$. We also added a regularization term λ to avoid overfitting. Similar to logistic regression, we used *Gradient Descent algorithm* to optimize the iteration process. Then, we take the partial derivatives of loss function w.r.t. weight and bias when classification is either correct or not. Then, we plug in these different values into the formula for updating parameters. We used tolerance, which is the difference between old and updated values, as an indicator of convergence and sign to stop.

Then, we applied the model on data before and after oversampling as a comparison. The run time for data is 0.09 and 4.5 seconds, respectively, with and without oversampling. Classification reports are shown below.

	0	1	accuracy	macro avg	weighted avg
precision	0.839590	0.0	0.83959	0.419795	0.704912
recall	1.000000	0.0	0.83959	0.500000	0.839590
f1-score	0.912801	0.0	0.83959	0.456401	0.766379
support	984.000000	188.0	0.83959	1172.000000	1172.000000

Table 7: Classification Report(Without oversampling)

	0	1	accuracy	macro avg	weighted avg
precision	0.861586	0.767750	0.808684	0.814668	0.815305
recall	0.741633	0.877579	0.808684	0.809606	0.808684
f1-score	0.797122	0.818999	0.808684	0.808061	0.807912
support	747.000000	727.000000	0.808684	1474.000000	1474.000000

Table 8: Classification Report(With oversampling)

Decision Tree

Decision tree algorithm is a supervised learning technique that is often used to solve both regression and classification problems. It starts from a root node and splits into a group of subnodes(or decision nodes). At the end, it will arrive at terminal nodes. A key value that determines this binary split process is called *Gini Index*, which takes values between 0 and 1.

$$\text{Gini Index} = 1 - \sum_{I=1}^{n} (P_i)^2, \tag{7}$$

where P_i indicates the probability of an observation being assigned to a class.[8] There are two main assumptions of the model: 1) All training data is in the root node. 2) Generally, the response variable is categorical.[9]

We called the `DecisionTreeClassifier()` function in the `sklearn` library to fit the training data and predict on test data. After repeating the process for 10 times, we get the results for the model performance.The run time for data is 0.07 and 0.11 seconds, respectively, with and without oversampling. Classification reports are shown below.

	0	1	accuracy	macro avg	weighted avg
precision	0.839590	0.0	0.83959	0.419795	0.704912
recall	1.000000	0.0	0.83959	0.500000	0.839590
f1-score	0.912801	0.0	0.83959	0.456401	0.766379
support	984.000000	188.0	0.83959	1172.000000	1172.000000

Table 9: Classification Report(Without oversampling)

	0	1	accuracy	macro avg	weighted avg
precision	0.996000	1.000000	0.997965	0.998000	0.997973
recall	1.000000	0.995873	0.997965	0.997937	0.997965
f1-score	0.997996	0.997932	0.997965	0.997964	0.997965
support	747.000000	727.000000	0.997965	1474.000000	1474.000000

Table 10: Classification Report(With oversampling)

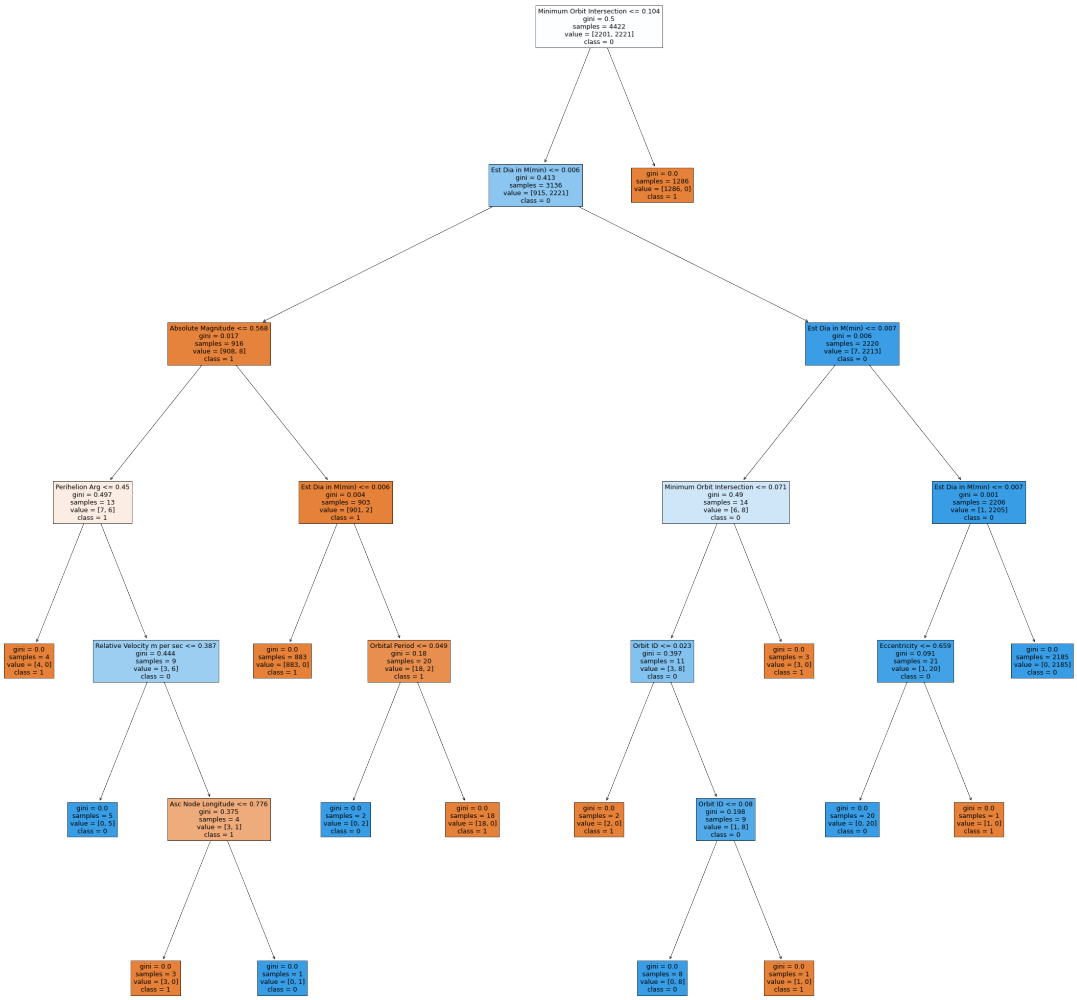

Figure 8: Decision tree plot(Without oversampling)

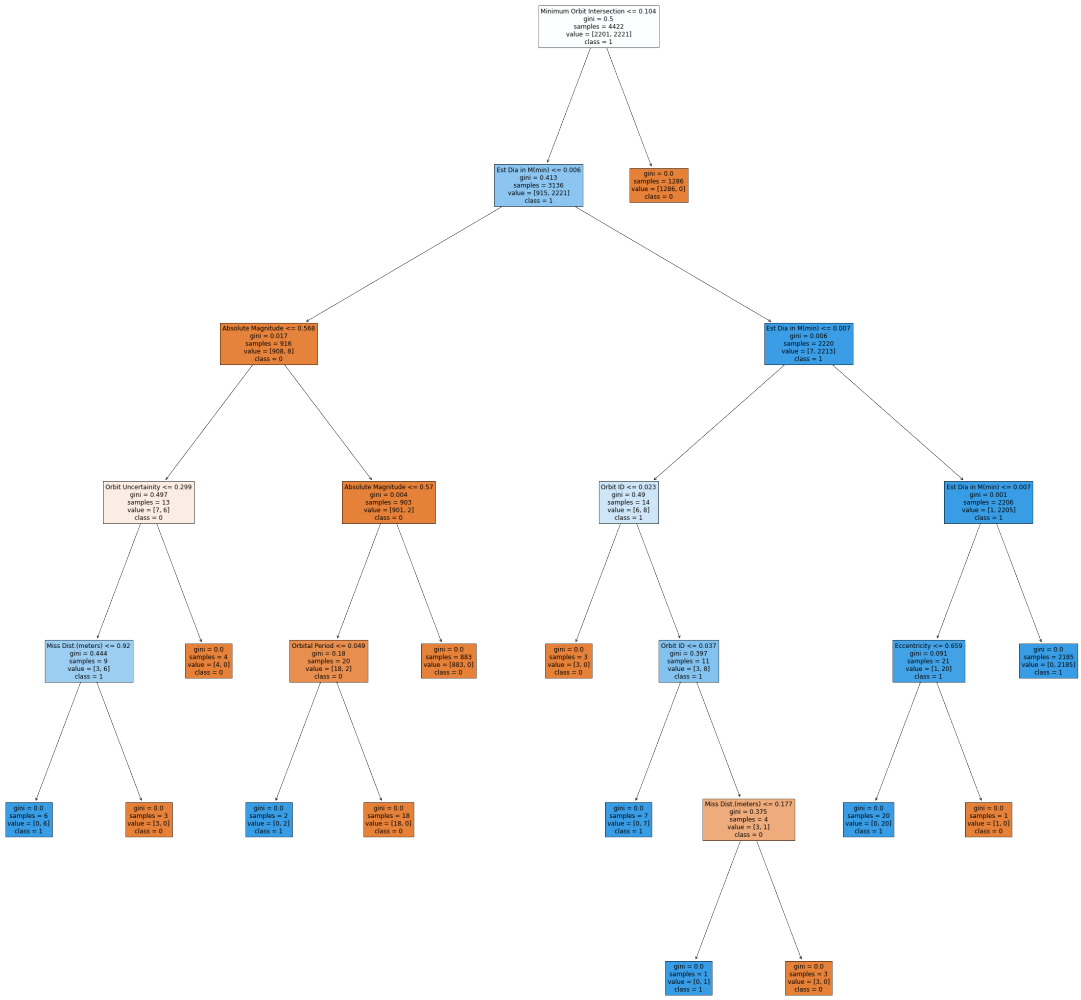

Figure 9: Decision tree plot(With oversampling)

Random Forest

Random Forest algorithm is a supervised learning technique that uses both bagging and decision tree methods. It only selects a subset of the possible feature split, which is different from *decision tree algorithm*. Therefore, we call it an ensemble model combining some of the decision trees to avoid overfitting.[10]

Before training, we need to set some hyperparameters, such as the number of trees or the number of features sampled. We called the `RandomForestClassifier()` function in the `sklearn` library with a maximum depth of two and set the random seeds. The run time is 0.54 and 1.89 seconds, respectively, with and without oversampling. Classification reports are shown below.

	0	1	accuracy	macro avg	weighted avg
precision	0.839590	0.0	0.83959	0.419795	0.704912
recall	1.000000	0.0	0.83959	0.500000	0.839590
f1-score	0.912801	0.0	0.83959	0.456401	0.766379
support	984.000000	188.0	0.83959	1172.000000	1172.000000

Table 11: Classification Report(Without oversampling)

	0	1	accuracy	macro avg	weighted avg
precision	0.997330	1.000000	0.998643	0.998665	0.998647
recall	1.000000	0.997249	0.998643	0.998624	0.998643
f1-score	0.998663	0.998623	0.998643	0.998643	0.998643
support	747.000000	727.000000	0.998643	1474.000000	1474.000000

Table 12: Classification Report(With oversampling)

K-nearest Neighbors

K-nearest Neighbors algorithm(k-NN) is also a non-parametric supervised learning method under assumption that data points from the same class are also close in distance.[11]

Before the implementation of the model, we first created a function that calculates the euclidean distance between two data points. Then, we called this function inside the model to compute the distance between all points in the test and training data. After sorting the observations by distance, we stored the indices of the k nearest neighbors and extracted their class labels. Lastly, we used the mode function to find the most common class.

Then, we applied the model on data before and after oversampling as a comparison. The run time for data is 79.39 and 222.17 seconds, respectively, with and without oversampling. Classification reports are shown below.

	0	1	accuracy	macro avg	weighted avg
precision	0.839590	0.0	0.83959	0.419795	0.704912
recall	1.000000	0.0	0.83959	0.500000	0.839590
f1-score	0.912801	0.0	0.83959	0.456401	0.766379
support	984.000000	188.0	0.83959	1172.000000	1172.000000

Table 13: Classification Report(Without oversampling)

	0	1	accuracy	macro avg	weighted avg
precision	0.986907	0.833140	0.896879	0.910023	0.911067
recall	0.807229	0.988996	0.896879	0.898112	0.896879
f1-score	0.888071	0.904403	0.896879	0.896237	0.896126
support	747.000000	727.000000	0.896879	1474.000000	1474.000000

Table 14: Classification Report(With oversampling)

XGBoost

XGBoost is a supervised learning algorithm from the CART(Classification And Regression Tree). It starts from ensemble models combining some of the decision trees. It is a block structure to support the parallelization of tree construction and prevent overfitting while still maintaining decent speed and model performance. Since we applied oversampling on our dataset, XGBoost works well in this situation to prevent overfitting and reduce the model's computation time.[12]

We called the `XGBClassifier()` function in the `xgboost` library with zero verbosity. The result shows that `xgboost` is efficient and has excellent prediction performance. The run time is 0.77 and 0.8 seconds, respectively, with and without oversampling. Classification reports are shown below.

	0	1	accuracy	macro avg	weighted avg
precision	0.839590	0.0	0.83959	0.419795	0.704912
recall	1.000000	0.0	0.83959	0.500000	0.839590
f1-score	0.912801	0.0	0.83959	0.456401	0.766379
support	984.000000	188.0	0.83959	1172.000000	1172.000000

Table 15: Classification Report(Without oversampling)

	0	1	accuracy	macro avg	weighted avg
precision	0.998663	1.000000	0.999322	0.999332	0.999322
recall	1.000000	0.998624	0.999322	0.999312	0.999322
f1-score	0.999331	0.999312	0.999322	0.999321	0.999322
support	747.000000	727.000000	0.999322	1474.000000	1474.000000

Table 16: Classification Report(With oversampling)

[13]

3 Conclusion and Limitations

LIMITATIONS:

- VIF
  As VIF is calculated using the R^2 values calculated from Multiple Regression, the imbalanced nature of the data might have affected some results. To compensate for that, a comprehensive check for variables was done when looking at the density plots.
- Ridge regression
  Ridge regression introduces the bias of variance trade-off model. This means that to reduce the variance, more bias is introduced in the estimators. It is not a reliable method for feature selection.
- Contingency Tables
  It is important to note that the binning of values into continuous intervals causes a loss of information and the interpretation does not provide a tangible explanation but provides a rather closer approximated explanation of the variables.
- Oversampling
  Using SMOTE method for oversampling has some drawbacks. It oversamples values that does not carry meaningful information. For modeling like Knn, oversampling makes it difficult to determine the number of nearest neighbors and is computationally expensive too as shown in the comparison table for the algorithms.

CONCLUSION:

Model comparison:

Comparison

Performance of each algorithm on data with/without oversampling

Algorithm	Accuracy	Run time (seconds)
Logistic Regression w/o oversampling	0.8396	1.3
Logistic Regression w/ oversampling	0.7517	2.02
Naive Bayes w/o oversampling	0.8396	0.21
Naive Bayes w/ oversampling	0.9423	0.12
Support Vector Machines w/o oversampling	0.8396	0.09
Support Vector Machines w/ oversampling	0.8087	4.5
Decision Tree w/o oversampling	0.8396	0.07
Decision Tree w/ oversampling	0.9980	0.11
Random Forest w/o oversampling	0.8396	0.54
Random Forest w/ oversampling	0.9986	1.89
K-nearest Neighbor w/o oversampling	0.8396	79.39
K-nearest Neighbor w/ oversampling	0.8969	222.17
XGBoost w/o oversampling	0.8396	0.77
XGBoost w/ oversampling	0.9993	0.8

Table 17: Comparison table of each model's performance

From the table, we could observe that the oversampling procedure increases the accuracy of all the models by a considerable amount except for *Logistic Regression* and *Support Vector Machine model*. In addition, it also result in much better prediction accuracy of the three models that we called directly from `sklearn` library, *Decision Tree, Random Forest, XGBoost*. By contrast, all seven models give almost the same prediction accuracy on the original data. One thing to also take note of is that *K-nearest Neighbors algorithm* takes exceptionally long time to run, which can be due to inefficient use of loops inside the algorithm. Based on the accuracy and time efficiency result of the seven models, we conclude that *Decision Tree algorithm* is the best model to use for this specific dataset.

Variables that explained the hazardous nature of the asteroid:

Reviewing our goal for the project and the hypothesis that we made in the introduction, the most important features that explain the hazardous nature of asteroids were Minimum Orbit Intersection, Absolute Magnitude, Estimated Diameter, Relative velocity, Orbit Uncertainty, Eccentricity and Perihelion Distance, Miss Distance as seen from ridge trace, density plot, contingency table and the decision tree.

References

[1] Wessel N. van Wieringen. *Lecture notes on ridge regression*. May 2021. URL: https://arxiv.org/pdf/1509.09169.pdf.

[2] Cornellius Yudha Wijaya. *5 smote techniques for oversampling your imbalance data*. Oct. 2021. URL: https://towardsdatascience.com/5-smote-techniques-for-oversampling-your-imbalance-data-b8155bdbe2b5.

[3] Jiang Z. et.al. *A New Oversampling Method Based on the Classification Contribution Degree*. 2021. URL: https://doi.org/10.3390/sym13020194.

[4] Zach. *The 6 assumptions of logistic regression (with examples)*. Oct. 2020. URL: https://www.statology.org/assumptions-of-logistic-regression/.

[5] *Learn naive Bayes algorithm: Naive Bayes classifier examples*. Aug. 2021. URL: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/.

[6] *Naive Bayes classifier*. Mar. 2022. URL: https://en.wikipedia.org/wiki/Naive_Bayes_classifier.

[7] Harry Davis. *Home*. Sept. 2019. URL: https://quick-adviser.com/what-are-the-assumptions-of-svm/.

[8] Neelam Tyagi. *Understanding the gini index and information gain in decision trees*. Sept. 2020. URL: https://medium.com/analytics-steps/understanding-the-gini-index-and-information-gain-in-decision-trees-ab4720518ba8.

[9] *Decision tree algorithm, explained*. URL: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html.

[10] IBM Cloud Education. *What is Random Forest?* URL: https://www.ibm.com/cloud/learn/random-forest.

[11] Onel Harrison. *Machine learning basics with the K-nearest neighbors algorithm*. July 2019. URL: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

[12] Jason Brownlee. *A gentle introduction to XGBoost for applied machine learning*. Feb. 2021. URL: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/.

[13] Python-Engineer. *MLfromscratch*. URL: https://github.com/python-engineer/MLfromscratch/tree/master/mlfromscratch.