Data Intensive Computing – Project Phase 2

REPORT

Team Members:
Kavitha Elizebeth Varughese
Meghana Gorripati
Tanvi Ranga

Problem: Identify the classification of asteroids and understand its proximity from other planets based on their perihelion point and aphelion point

1. Model: K Nearest Neighbors.

We chose perihelion point and aphelion point for classification based on our EDA. The graph on the right that we got after our EDA, tells us that there is a direct correlation between the classification of asteroids and their perihelion and aphelion points.
With the classification, we can also tell how far the asteroid will be from the planets. We decided on the KNN algorithm as the classification for the clusters is well defined. The clusters are well segregated, and we have enough data for supervised training and testing.

The k-nearest neighbors (K-NN) algorithm is a simple, yet effective supervised machine learning algorithm used for classification and regression tasks. Mathematically, K-NN can be described as follows:

Given a dataset D = $\{(x_1, y_1), (x_2, y_2) \ldots\ldots (x_n, y_n)\}$,
where,

x_i represents the features of the ith data point.

y_i represents the target of the ith data point.

x_{new} for which we want to predict the target values

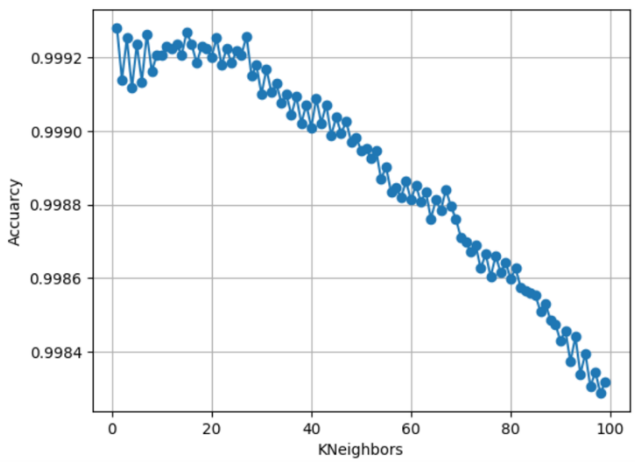
K-NN algorithm works as follows:

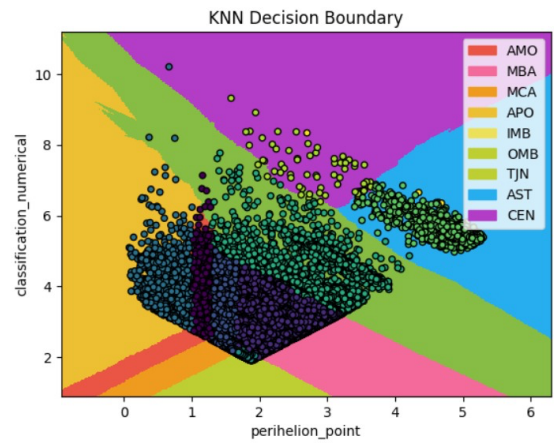
1. Calculate the distance of new data points from all the data points in the training set.
2. Compute the distance between the new data point x_{new} and all other data points in the dataset. Common distance metrics include Euclidean distance, Manhattan distance, or cosine similarity.
   Euclidean distance between two points p and q in n-dimensional space is calculated as,

$$d_{(p,q)} = \sqrt{\Sigma_{i=1}^{n} (p_i - q_i)^2}$$

3. Find the k-nearest neighbor: Select the k-data points from the set which are closest to x_{new} , based on the calculated distances.
4. For classification purposes, many of the nearest neighbors will decide the class for x_{new}, whereas for regression purposes, x_{new} is the average of the target values of k-nearest neighbors.

We have divided our data with 70:30 ratio with 70% of data taken for training and 30% for testing.





To decide on the value of K, we experimented with multiple values of K and their corresponding accuracies, resulting to the graph on the top left. In this graph, we can see that the accuracy has comparatively large variations for values less than K=10. And after K=25, the accuracy decreases on average.

So, we decided to take K = 25.

With K = 25 and with 9 classifications, the decision boundary is on the top right and the metrics is below.

```
Accuracy: 0.9992177519654686
Precision metric: 0.8735
Recall metric: 0.8578
F1 metric: 0.8652
```

Problem: Identify whether the asteroids are potentially hazardous or not.

Data Preprocessing

From the EDA, we have observed the data is highly imbalanced with over 99% of the data belonging to the negative class. This could sway the models and predict only the negative class for any input. For this reason, its best to oversample the positive class and create an equal sample number for both classes. This is achieved by using the library SMOTE.

The data needs to be normalized before using it to train models, so all the numeric features need to be on the same scale. For this we have used min-max scaler.

	Importance
perihelion_point	0.792928
absolute_magnitude	0.144963
mean_motion	0.050902
inclination	0.011207

From the EDA and correlation matrix we have discovered the features which are impacting the classification of asteroid as potentially hazardous or not are absolute magnitude, perihelion point, inclination, and mean motion, eccentricity, and semi major axis.

Since perihelion point is equally correlated with eccentricity and semi major axis, from the principal component analysis for dimensionality reduction we will consider the features. Absolute magnitude, perihelion point, inclination, and mean motion to classify the asteroids as potentially hazardous or not.

1. Model: Logistic Regression

Logistic regression is particularly useful when the outcome or dependent variables is binary (i.e., the target column has two values 0 and 1), making it well suited for binary classification problems.

The coefficients in logistic regression models represent the relationship between the independent variables and the log-odds of the outcome. This makes logistic regression models easy to interpret compared to more complex models like neural networks.

Logistic regression models output probabilities, which can be interpreted as the likelihood of a particular outcome occurring. This probabilistic interpretation is valuable in many applications, especially when decision-making involves assessing risks and uncertainties.

The probabilities are given by sigmoid functions.

$$f(x) = \frac{1}{1 + e^{-x}}$$

As per the correlation matrix we plotted during the EDA phase, we know that there are some prominent features which will affect the decision of the model more than others.

Features such as absolute magnitude, semi-major axis, perihelion point, aphelion point play a significant role in predicting whether an asteroid is potentially hazardous or not.

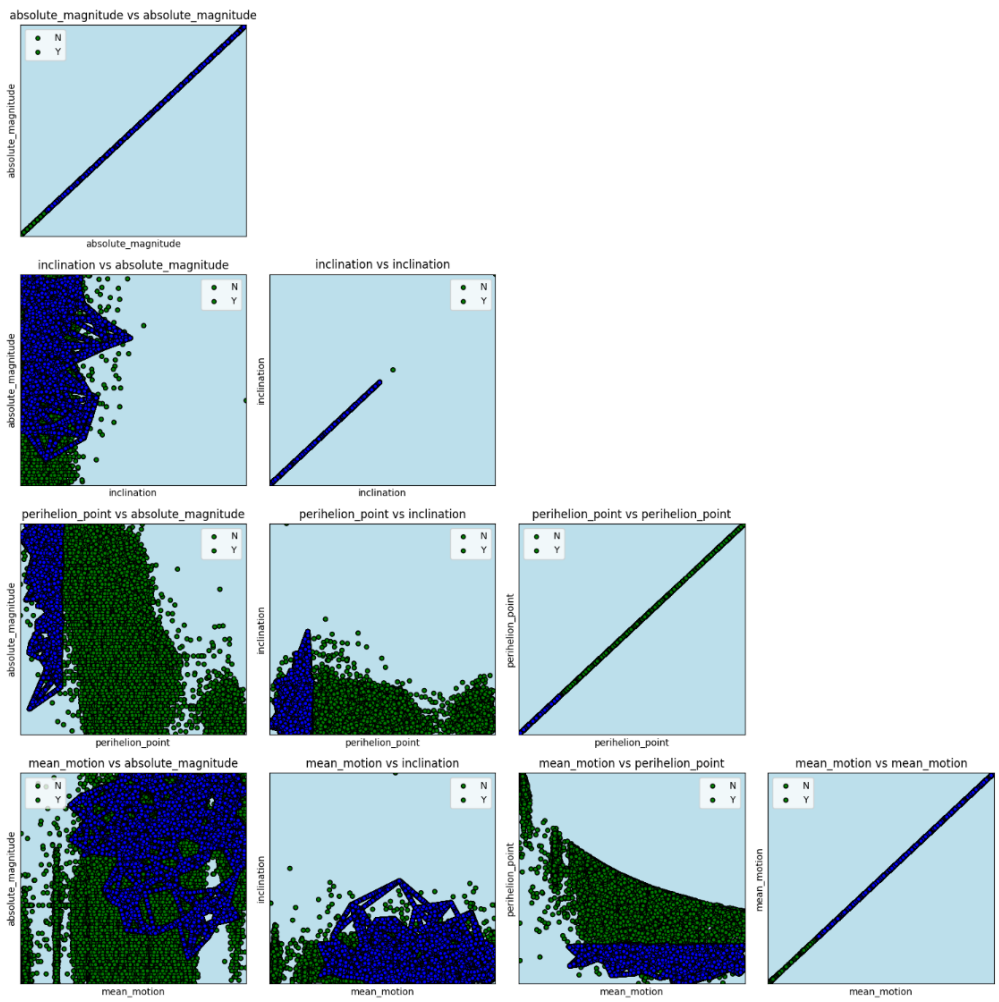
We have implemented Logistic Regression with features such as absolute magnitude, semi-major axis, perihelion point, and mean motion. The screenshot of the metrics measurement for the model can be seen below.

```
Accuracy: 0.9970811025778592
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    266404
           1       0.15      1.00      0.26       135

    accuracy                           1.00    266539
   macro avg       0.57      1.00      0.63    266539
weighted avg       1.00      1.00      1.00    266539
```

```
Confusion Matrix:
         0    1
0   265626  778
1        0  135
```

The potentially hazardous asteroids classes are encoded as 1 and non-hazardous asteroids are encoded as 0. The accuracy with the selected features for Logistic Regression model is 99.7 %.

The decision boundary plot for the Logistic Regression is:



From the above decision boundaries plot we can see that the data is highly dimensional and not clearly linearly separable. To improve the efficiency and performance of the application we are implementing the Support Vector Machines, since SVM uses Gaussian Kernels to deal with nonlinear data and it uses extra margin width near to decision boundary to deal with support vectors (points which lie near to the decision boundary and affects the results most).

2. Model: Support Vector Machine

We have already implemented the Logistic Regression for identification of potentially hazardous asteroids. Since Logistic Regression has some limitations such as:

Standard logistic regression is susceptible to noise data.

Logistic Regressions assumes the linear relationship between data and the target which may limit accuracy for the non-linear data.

We have chosen Support Vector Machines since Support Vector Machines are less sensitive to outliers and aims to maximize the margin between decision boundaries and the closest data points (support vectors).

Since we have already discovered the effect of absolute magnitude, semi major axis, perihelion point, and mean motion on the decision of models.

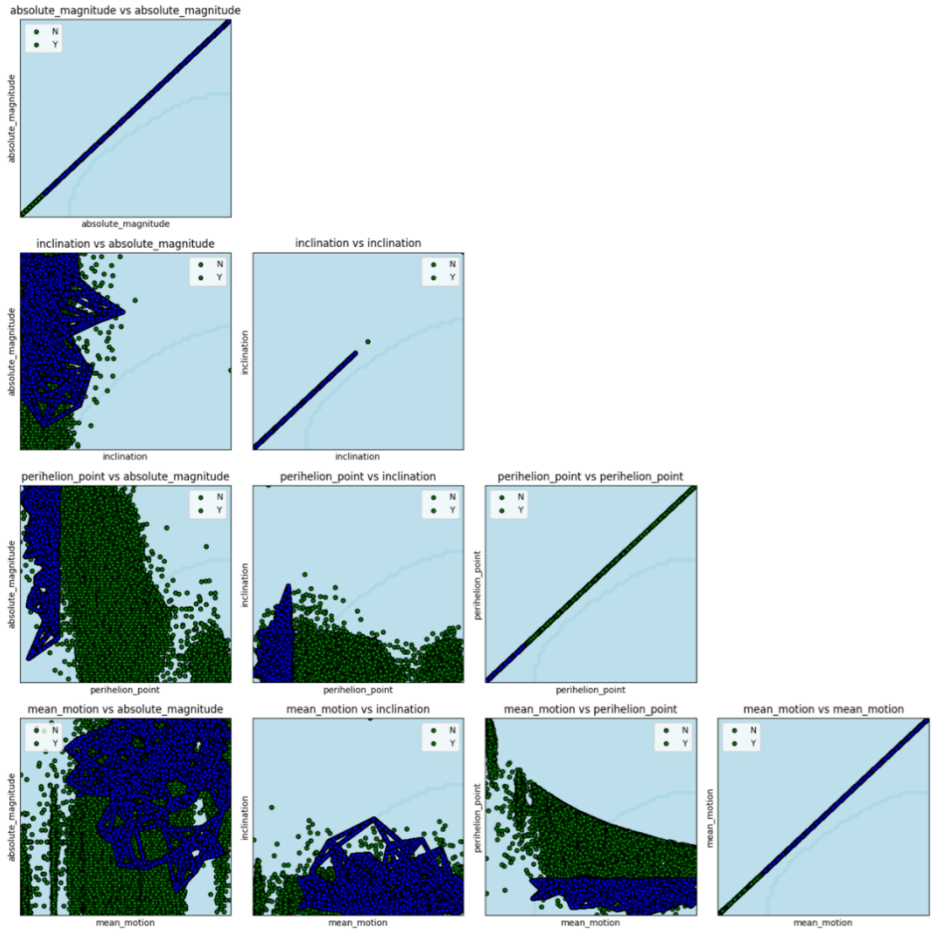
We have fitted the Support Vector Machines (SVM) using the mentioned four features. The results are as seen on the right.

```
Accuracy: 0.9982779255568603
Precision metric: 0.6136
Recall metric: 0.9991
F1 metric: 0.6848
```

```
Confusion Matrix:
        0    1
0  265945  459
1       0  135
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    266404
           1       0.23      1.00      0.37       135

    accuracy                           1.00    266539
   macro avg       0.61      1.00      0.68    266539
weighted avg       1.00      1.00      1.00    266539
```

The decision boundary plot for the model is shown below,



We can clearly see that the data is not linearly separable, hence it's computationally expensive and not easily interpretable from Support Vector Machines (SVM) so we are implementing decision trees, ensembles and gradient boosting as decision trees are computationally less expensive, faster in execution and easier to visualize and interpret the results.

3. Model: Decision Trees

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

Some notable advantages of decision trees are:

1. Decision trees are generally easier to interpret and understand than SVMs. Decision trees provide a clear, intuitive representation of the decision-making process, as they can be visualized and easily explained. Each branch of the tree corresponds to a decision based on a feature, making it straightforward to trace how the model arrives at a prediction.
2. Decision trees inherently capture non-linear relationships between features and the target variable without requiring explicit transformation of the data. They recursively partition the feature space, creating piecewise linear decision boundaries, which can effectively model complex decision boundaries.
3. Decision trees tend to be computationally less expensive to train compared to SVMs, especially for large datasets. SVMs involve solving a convex optimization problem, which can be computationally intensive, particularly when dealing with many training samples or high-dimensional feature spaces.

4. Decision trees are generally robust to noisy data and outliers. They can handle noisy datasets or datasets with missing values without significant preprocessing. In contrast, SVMs can be sensitive to outliers, as they aim to maximize the margin between classes, which can lead to overfitting if outliers are not properly addressed.
5. Decision trees provide a natural way to assess the importance of features in the prediction process. Features that appear higher in the tree and are closer to the root node are typically more important in making predictions. This can be valuable for feature selection and understanding which features are driving the model's decisions.

How Decision Tree works:

A decision tree is a popular supervised machine learning algorithm used for classification and regression tasks. It works by recursively partitioning the feature space into smaller regions, each associated with a particular class or predicted value. Here's a brief explanation of how decision trees work, along with some mathematical concepts involved:

Splitting Criteria:

At each node of the tree, the algorithm selects the feature and the split point that best separates the data into different classes or groups. This process is done by evaluating different splitting criteria, commonly using metrics like Gini impurity or entropy for classification and mean squared error for regression.

Tree Structure:

The decision tree starts with a root node that contains the entire dataset. Then, it recursively splits the data into smaller subsets based on the selected features and split points, creating a tree-like structure.

The splitting process continues until a stopping criterion is met, such as reaching a maximum tree depth, having minimum samples in a node, or when no further improvement in purity or error reduction is possible.

Decision Rules:

Each path from the root node to a leaf node in the decision tree represents a decision rule or a series of conditions that lead to a particular outcome.

For example, if a decision tree is trained to classify whether an email is spam or not, a decision rule could be: "If the email contains the word 'free' and the sender is not in the contact list, classify it as spam."

Prediction:

To make predictions for new instances, the algorithm traverses the decision tree based on the feature values of the instance, following the decision rules learned during training.

When it reaches a leaf node, the prediction is made based on the majority class in that node (for classification) or the average value (for regression).

Mathematically, decision trees use algorithms to determine the best split points based on certain criteria. For example, in classification tasks, the Gini impurity or entropy measures are used to quantify the homogeneity of classes in each node. The algorithm aims to minimize these impurity measures by selecting the split that maximally separates the classes. In regression tasks, the algorithm minimizes the mean squared error by selecting splits that best partition the data into regions with similar response values.

The entropy of a given data set is defined as

$$H(S) = \Sigma_{i=1}^{n} p_i \log_2(p_i)$$

Where, p_i is the proportion of instances of class c_i in the set.

Information gain measures the reduction in entropy achieved by splitting a set of data on a particular feature. It helps the decision tree algorithm decide which feature to choose for splitting at each node.

The information gain is given by,

$$IG(D, A) = H(D) - \Sigma_{v \in Values(A)} \frac{|D|}{|D_v|} H(Dv),$$

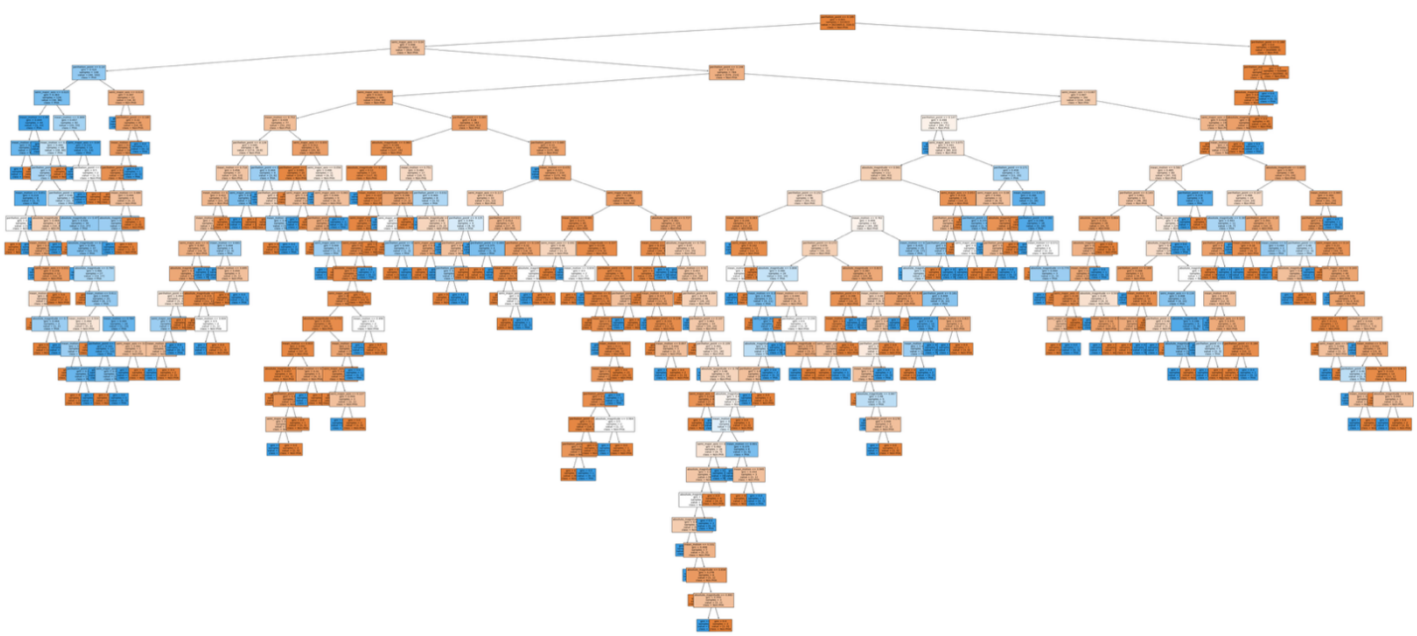
where

- .H(D) is the entropy of the dataset before splitting.
- $|D_v|$ is the number of instances in dataset D with values ***v*** for feature A.
- $|D|$ is the total number of instances in D.
- Values(A) represent the possible values of feature A.
- $H(D_v)$ is the entropy of subset D with values v for feature A.

From the results of Logistic Regression and Support Vector Machines, we have already discovered the principle features as absolute magnitude, semi major axis, perihelion point and aphelion point.

We have fitted Decisions Trees on absolute magnitude, semi major axis, perihelion point and aphelion point.

The tree plot as



First Split based on entropy.

The results for the model are as shown in the metrics on the left and the right.

```
Precision metric: 0.7499
Recall Metric: 0.7702
Accuracy Metric: 0.9995
F1 score: 0.7597
```

```
Confusion Matrix:
          0   1
0  266310  94
1      82  53
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    266404
           1       0.37      0.87      0.52       135

    accuracy                           1.00    266539
   macro avg       0.69      0.94      0.76    266539
weighted avg       1.00      1.00      1.00    266539
```

4. Model: Random Forest

Decision Trees serve as a fundamental tool in predictive modeling, offering interpretability and simplicity. However, they suffer from limitations such as overfitting and instability. Random Forest, an ensemble method, addresses these drawbacks while enhancing predictive performance.

The rationale behind choosing Random Forest Algorithms after implementing Decision Trees based on several reasons:

a. Addressing Overfitting:

Decision Trees are prone to overfitting, especially with complex datasets or deep trees. Random Forest mitigates this issue by constructing multiple trees on bootstrapped samples of the dataset and aggregating their predictions through voting or averaging. This ensemble approach reduces variance and improves generalization performance.

b. Scalability and Efficiency:

Random Forest exhibits favorable scalability characteristics compared to individual Decision Trees, as the training process can be parallelized across multiple processors or distributed computing clusters. This scalability enhances efficiency, making Random Forest suitable for handling large datasets and real-time prediction tasks.

c. Handling High-Dimensional Data:

In high-dimensional feature spaces, Decision Trees may struggle to capture relevant patterns effectively, leading to suboptimal performance. Random Forest can handle such scenarios more robustly by randomly selecting subsets of features for each tree. This feature randomness reduces correlation among trees and encourages diverse tree structures, thereby improving the model's overall predictive power.

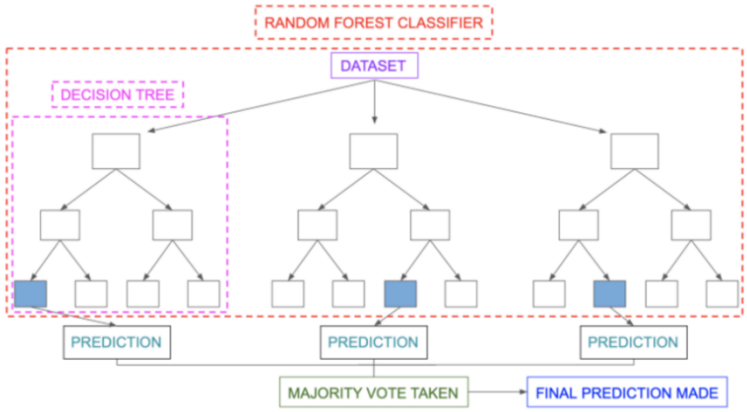
Steps involved in Random Forest Algorithm:

Step-1 – We first make subsets of our original data. We will do row sampling and feature sampling that means we'll select rows and columns with replacement and create subsets of the training dataset.

Step- 2 – We create an individual decision tree for each subset we take.

Step-3 – Each decision tree will give an output.

Step 4 – Final output is considered based on Majority Voting if it's a classification problem and average if it's a regression problem.



Mathematically, if we denote the random forest as $RF(x)$ for a given input vector *x,* and each individual tree as $T_i(x)$ where i =1,2,3...n (where n is the number of trees in the forest), the prediction of random forest can be explained as

$$RF(x) = mode\big(T1(x), T2(x), \dots, Tn(x)\big) \qquad \text{... for classification.}$$

$$RF(x) = \frac{1}{n}\Sigma_{i=1}^{n} T_i(x) \qquad \text{... for regression.}$$

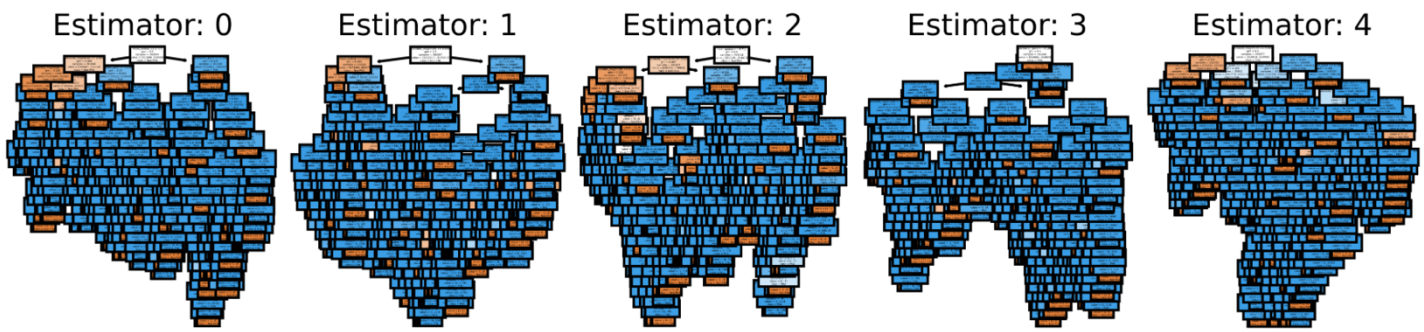
We have fitted our dataset in Random Forest model with our data columns as absolute magnitude, semi major axis, perihelion, and mean motion.

The Random Forest Plot as well the metrics results are as shown below.

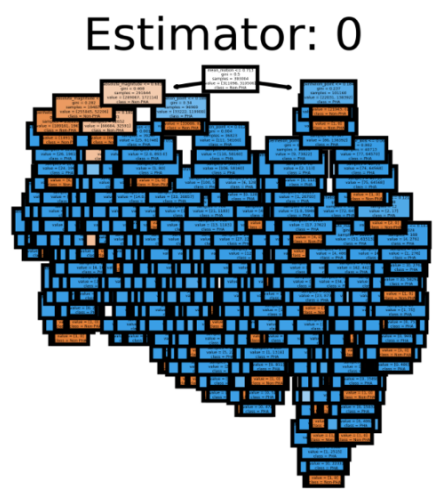
```
Precision metric: 0.6840429443797827
Recall Metric: 0.9699612173546499
Accuracy Metric: 0.9991520940650336
F1 score: 0.7643711658073569
```

```
Confusion Matrix:
[[266186    218]
 [     8    127]]
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    266404
           1       0.37      0.94      0.53       135

    accuracy                           1.00    266539
   macro avg       0.68      0.97      0.76    266539
weighted avg       1.00      1.00      1.00    266539
```



We are visualizing the first five estimators of the Random Forest Algorithms with the five trees created using bootstrapping and bagging of the dataset.



As Random Forest will randomly select the data and build a tree, it makes the algorithm more robust and less sensitive to training data.

Since the estimator 0 is having mean_motion, estimator1 is having absolute magnitude, estimator 2 is having mean motion, estimator 3 is having perihelion point and estimator 4 is having absolute magnitude as the prominent feature of first split (root node). Unlike Decision Trees, Random Forest trains the data with all the possible combinations of features.

5. Model: XGBoost Classifier

XGBoost is one of the most popular and efficient implementations of the Gradient Boosted Trees algorithm, a supervised learning method that is based on function approximation by optimizing specific loss functions as well as applying several regularization techniques.

The objective function (loss function and regularization) at iteration ***t*** that we need to minimize is the following:

Real value (label) known from the training data-set

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

Can be seen as f(x + Δx) where x = $\hat{y}_i^{(t-1)}$

Mathematically, XGBoost can be explained as follows:

Let's assume we have a dataset with N rows and D features represented as (x_i, y_i), where xi is the feature vector for sample i and y_i is the corresponding target value. XGBoost aims to learn an additive model F(x) that minimize a loss function $L(y,\ F(x))$ where y represents the true target values.

The additive function can be represented as

$$F(x) \ = \ \Sigma_{m=1}^{M} f_m(x),$$

where

M is the number of base learners(trees) in the ensemble.

$f_m(x)$ represents the mth base learner(tree)

The objective of XGBoost is to find the optimal parameters Θ for each base learner $f_m(x)$ by minimizing the following objective function.

$$Obj(\theta) = \Sigma_{i=1}^{N} L(y, y_i^{\wedge}) + \Sigma_{m=1}^{M} \Omega(f_m),$$

where

$L(y, y_i^{\wedge})$ is the loss function measuring the true target value y_i and the predicted value.

$\Omega(f_m)$ is a regularization term penalizing the complexity of the mth tree, which helps prevent overfitting.

Θ represents the parameters of the base learner $f_m(x)$ (e.g., tree structure, leaf values).

XGBoost optimizes this objective function through a gradient-based optimization technique. It iteratively adds new base learners to the ensemble by fitting them to the negative gradient of the loss function with respect to the ensemble predictions.

During training, XGBoost builds trees sequentially, where each new tree is trained to correct the errors (residuals) of the previous ensemble predictions. This is achieved by calculating the negative gradient of the loss function with respect to the ensemble predictions and using it as the target for the new tree.

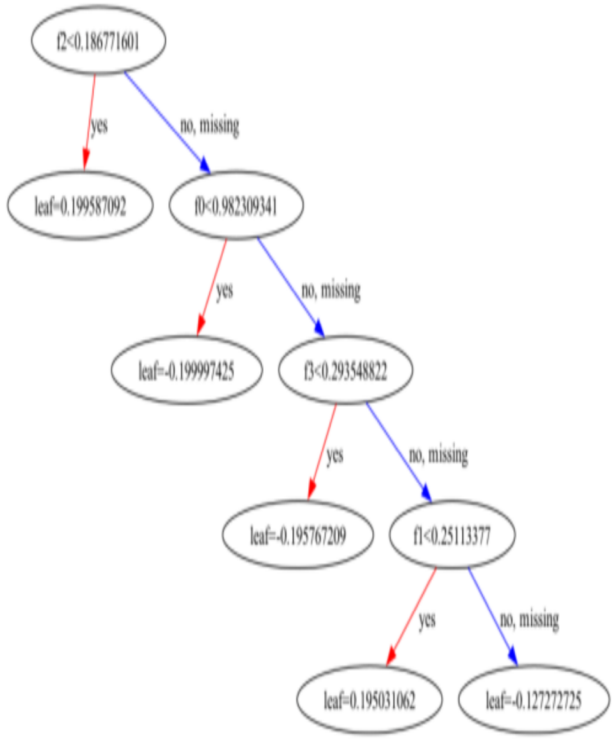
The results of the metrics and the related graphs are as shown below.

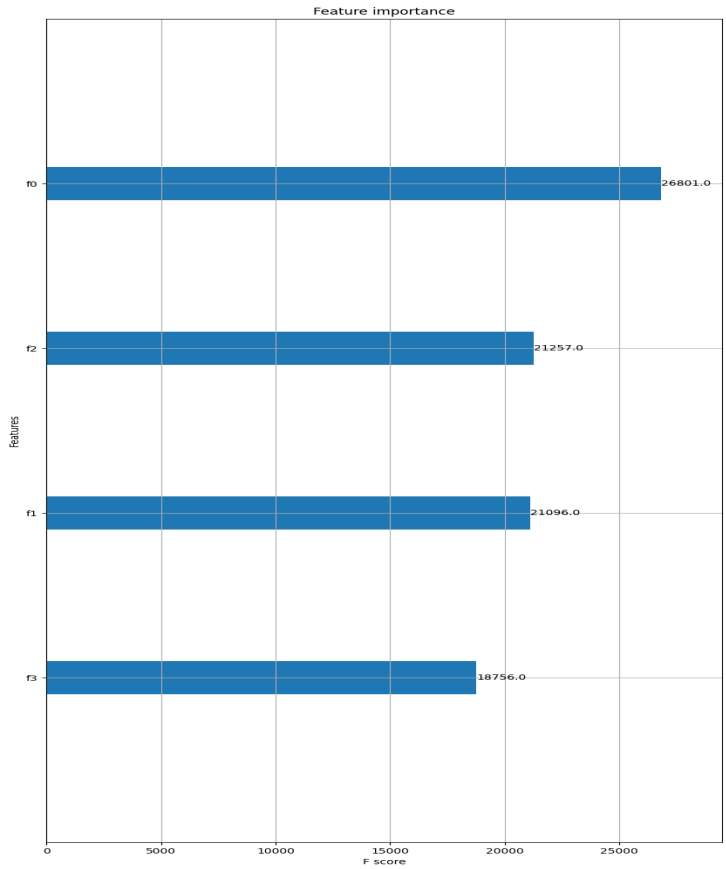
```
Precision metric: 0.6959726288716328
Recall Metric: 0.9700006311772651
Accuracy Metric: 0.9992308817846544
F1 score: 0.7764960079026761
```

```
Confusion Matrix:
[[266207    197]
 [     8    127]]
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    266404
           1       0.35      0.86      0.50       135

    accuracy                           1.00    266539
   macro avg       0.68      0.93      0.75    266539
weighted avg       1.00      1.00      1.00    266539
```

The most important feature as per the XGBoost feature importance plot is absolute magnitude. The tree has root node as Perihelion point as per the maximum information gain.





Problem Synopsis

Accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

Precision: The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

Recall: The recall is intuitively the ability of the classifier to find all the positive samples.

F1 score: The F1 score can be interpreted as a harmonic mean of the precision and recall.

Confusion Matrix: a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i and predicted to be in group j.

We have compared the correctness metrics such as Precision, Recall and F1 score for all the 5 implemented classifiers: Logistic Regression, SVM, Decision Tree, Random Forest, XGBoost.

	Model	Precision	Recall	F1 Score
1	Logistical Regression	0.573932	0.998540	0.628086
2	SVM	0.613636	0.999139	0.684754
3	Decision Tree	0.767016	0.788761	0.777463
4	Random Forest	0.684043	0.969961	0.764371
5	XGBoost	0.695973	0.970001	0.776496

```
MEDIAN VALUES

Precision    0.684043
Recall       0.970001
F1 Score     0.764371
dtype: float64
```

```
BEST PRECISION MODEL

Model        Decision Tree
Precision         0.767016
Recall            0.788761
F1 Score          0.777463
Name: 3, dtype: object
```

```
BEST RECALL MODEL

Model             SVM
Precision    0.613636
Recall       0.999139
F1 Score     0.684754
Name: 2, dtype: object
```

```
BEST F1 MODEL

Model        Decision Tree
Precision         0.767016
Recall            0.788761
F1 Score          0.777463
Name: 3, dtype: object
```

Problem: Predict the position of asteroids where they are nearest and the farthest away from the sun

Perihelion and aphelion are terms used in astronomy to describe the points in an orbit around the Sun where an object, such as an asteroid, is closest to and farthest from the Sun, respectively.

Perihelion: This refers to the point in an asteroid's orbit around the Sun where it is closest to the Sun. At perihelion, the asteroid is traveling at its highest velocity due to the gravitational pull of the Sun, and it experiences its highest temperature because of the proximity to the Sun's radiation.

Aphelion: This refers to the point in an asteroid's orbit around the Sun where it is farthest from the Sun. At aphelion, the asteroid is traveling at its slowest velocity in its orbit, and it experiences its lowest temperature because of the distance from the Sun.

For many asteroids, their orbits are elliptical, meaning they are not perfect circles but rather elongated shapes. Therefore, the distance between the asteroid and the Sun varies throughout its orbit, and perihelion and aphelion represent the extremes of this distance.

To calculate the perihelion and aphelion points of an asteroid's orbit, you need to know some key parameters of its orbit. These parameters include:

Semi-major axis (a): This is one-half of the longest diameter of the elliptical orbit. It represents the average distance between the asteroid and the Sun.

Eccentricity (e): This parameter describes how elongated the orbit is. It is a number between 0 and 1, where 0 represents a circular orbit and 1 represents a highly elongated orbit.

With these parameters, you can use the following formulas:

1. **Perihelion distance (q)**: $q = a \times (1 - e)$
2. **Aphelion distance (Q)**: $Q = a \times (1 + e)$

Data Preprocessing

From the EDA, we have observed the data is highly imbalanced with over 99% of the data belonging to the negative class. This could sway the models and predict only the negative class for any input. For this reason, its best to oversample the positive class and create an equal sample number for both classes. This is achieved by using the library SMOTE.

The data needs to be normalized before using it to train models, so all the numeric features need to be on the same scale. For this we have used min-max scaler.

1. Model: Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable (often denoted as y) and one or more independent variables often denoted as $x_1, x_2, \dots, x_n$. The relationship is assumed to be linear, meaning that the dependent variable is a linear function of independent variable.

The basic form of linear regression with one independent variable is expressed as:

$$y = mx + b$$

Where:

- y is the dependent variable (the variable we are trying to predict).
- x is the independent variable (the variable we are using to make predictions).
- m is the slope of the line (the change in y for a one-unit change in x).
- b is the y-intercept (the value of y when x is 0).

The goal of linear regression is to estimate the parameters m and b such that the line fits the data points as closely as possible. This is typically done by minimizing the sum of the squared differences between the observed values of y and the values predicted by the linear model.

Mathematically, for a dataset with n observations, the linear regression model can be represented as:

$$y_i = mx_i + b + \epsilon_i$$

where

y_i is the observed value of dependent variable for the ith observation

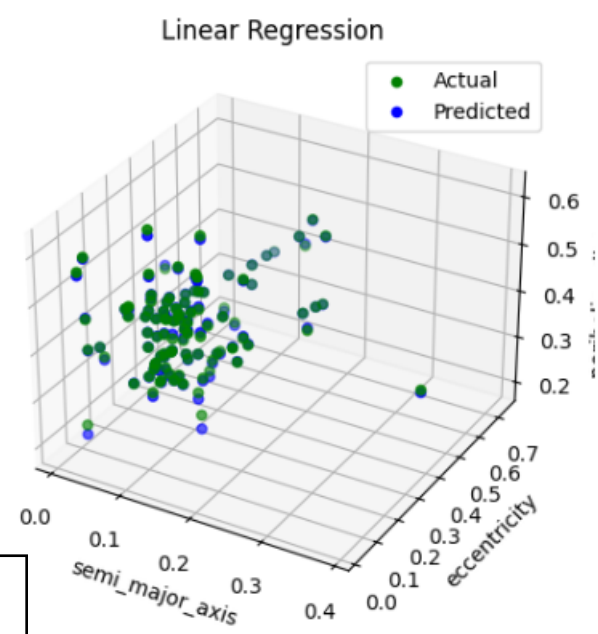
x_i is the observed value of independent variable for the ith observation.

ϵ_i is the error term for the ith observation, representing the difference between the observed values and the predicted values.

The objective is to minimize the loss function which is given by

$$SSE = \Sigma_{i=1}^{n} \left(y_i - (mx_i + b)\right)^2$$

We have fitted the Linear Regression Model to predict the perihelion distances using semi major axis (a) and eccentricity (e).
The plot depicts the 3-D plot between semi-major axis, eccentricity, and perihelion distances for 100 asteroids. Here, the green dots point to the actual values whereas blue dots point to the predicted value. This indicates a close alignment between the predicted and actual perihelion distances, as evidenced by a mean squared error of 3.87388e-05.



The Result metrics are:

```
Mean Absolute Error (MAE): 0.003368341138361751
Mean Squared Error (MSE): 3.873883587208452e-05
Root Mean Squared Error (RMSE): 0.0062240530100638215
R-squared (R^2): 0.9939652820974654
```

Conclusion:

In this project, we have tried to solve three problems with our dataset about Asteroids from NASA, with our limited knowledge on the domain and machine learning. We have applied KNN to classify asteroids to the classification defined by NASA based on its proximity from the sun. The proximity from the sun is found from the perihelion and aphelion points of the asteroids. We have applied five classification models, namely logistical regression, SVM, decision tree, random forest classifier and XGBoost, to classify asteroids into potentially hazardous or not, i.e. whether it would collide Earth to cause a catastrophic damage to Earth. Lastly, we used linear regression to predict the proximity of asteroids from sun by predicting the perihelion points and aphelion points based on the eccentricity and semi-major axis.

References:

1. https://matplotlib.org/3.1.1/index.html
2. An asteroid the size of a bus zoomed past Earth Wednesday — how close it get? (nypost.com)
3. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html
4. https://xgboost.readthedocs.io/en/stable/python/python_api.html
5. https://matplotlib.org/stable/index.html
6. https://www.geeksforgeeks.org/change-the-legend-position-in-matplotlib/
7. https://www.geeksforgeeks.org/matplotlib-axes-axes-get_legend_handles_labels-in-python/
8. https://scikit-learn.org/0.24/modules/generated/sklearn.neighbors.DistanceMetric.html
9. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
10. https://scikit-learn.org/stable/modules/svm.html
11. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
12. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
13. https://xgboost.readthedocs.io/en/stable/
14. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
15. https://arxiv.org/pdf/1603.02754.pdf
16. https://github.com/dmlc/xgboost)

- Please write the names of your group members.

Group member 1 : Kavitha Elizebeth Varughese

Group member 2 : Meghana Gorripati

Group member 3 : Tanvi Ranga

- Rate each groupmate on a scale of 5 on the following points, with 5 being HIGHEST and 1 being LOWEST.

Evaluation Criteria	Group member 1	Group member 2	Group member 3
How effectively did your group mate work with you?	5	5	5
Contribution in writing the report	5	5	5
Demonstrates a cooperative and supportive attitude.	5	5	5
Contributes significantly to the success of the project .	5	5	5
TOTAL	20	20	20

Also please state the overall contribution of your teammate in percentage below, with total of all the three members accounting for 100% (33.33+33.33+33.33 ~ 100%) :

Group member 1 : 33.33

Group member 2 : 33.33

Group member 3 : 33.33