Predicting Hazardous Near-Earth-Objects

Project phase - 2

50461788 - Sonalira
50471077 - Khyathik

Problem statement:

The objective of this project is to examine the dataset of near-Earth objects using data-driven analysis. This analysis aids in determining the likelihood of the incoming object, whether it is harmful and enters the earth's atmosphere.

Feature engineering:

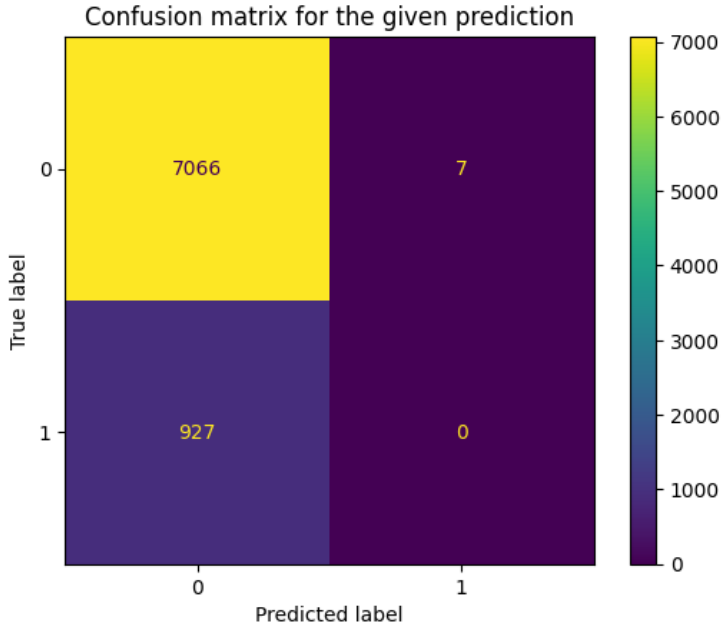
The process of converting unprocessed data into features that may be used to better understand our model and boost its accuracy.

We have performed feature engineering i.e., converting our data into train and test data sets to get a better understanding of data.

The machine learning models we used to extract findings from the dataset are listed below.

Logistic regression:

We used a Logistic regression model to predict the response variable as it is a binary response(0 or 1). Logistic regression was used to analyze the relationship between average estimated diameter, relative velocity, absolute magnitude, and [hazardous]. Logistic regression works better for models with large datasets when compared to Naive Bayes classifiers.
If we hold all the other predictor variables constant, the odds of an observed asteroid being hazardous increased with an increase in absolute magnitude

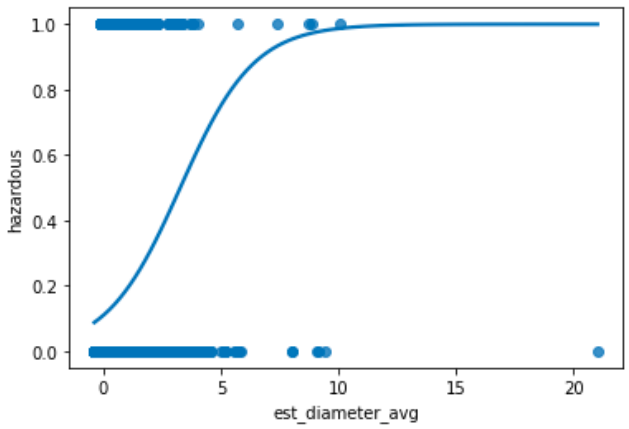


```
Accuracy: 0.88325
f1_score: 0.0
Precision: 0.0
Recall: 0.0
Classification report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.94      7073
           1       0.00      0.00      0.00       927

    accuracy                           0.88      8000
   macro avg       0.44      0.50      0.47      8000
weighted avg       0.78      0.88      0.83      8000
```

The accuracy of logistic regression classifier on the given dataset is 88.32%
From the above confusion matrix, we observe that there are 7066 true positives and 0 true negatives. 927 false positives and 7 false negatives.
Precision which is the ability of the classifier to not label a sample as positive if it is negative is 88%
Recall which is the ability of the classifier to find all the positive samples is 100%
Support which is the number of occurrences of each class in y_test is 7073.
We observe that accuracy using the logistic regression model is lower and can be improved. But if we observe, number of predictions (of 1's) is 0 (from confusion matrix). Thus, logistic regression is not a better fit for this problem.

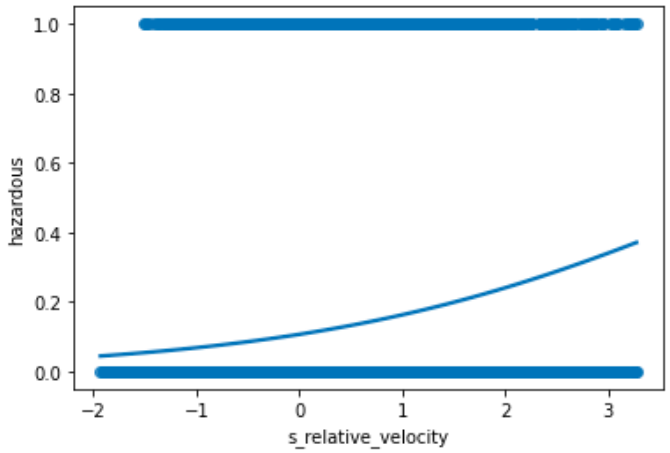
Visualizations:

```
: sns.regplot(x=s_x_test['est_diameter_avg'], y=y_test, logistic=True, ci=None)
: <AxesSubplot:xlabel='est_diameter_avg', ylabel='hazardous'>
```



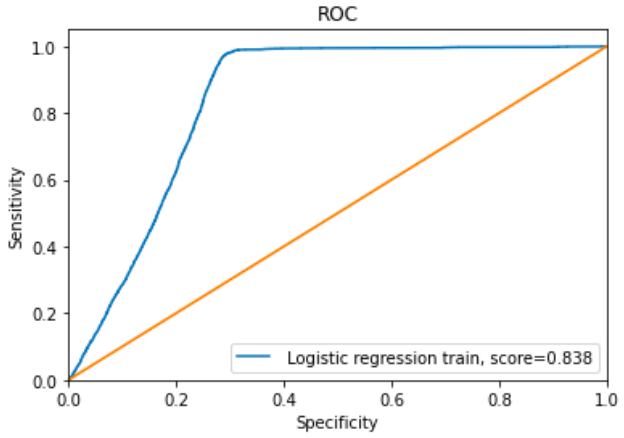
We observe that as the estimated diameter increases most of the asteroids are classified as hazardous.

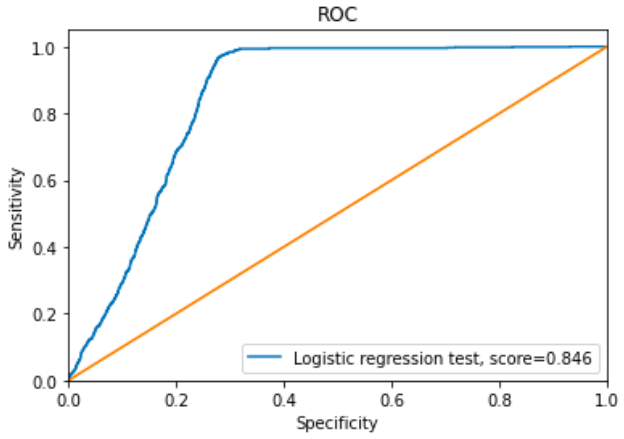
```
: sns.regplot(x=s_x_test['s_relative_velocity'], y=y_test, logistic=True, ci=None)
: <AxesSubplot:xlabel='s_relative_velocity', ylabel='hazardous'>
```



We observe that there is no high correlation between relative velocity and the hazardous nature of an asteroid.

ROC Curve:





Here, we used ROC curve to visualise sensitivity and specificity the two metrics to see how well a model fits a dataset. In the ROC curve, pairs of the true positive rate are plotted against the false positive rate for every possible decision threshold of a logistic regression model.

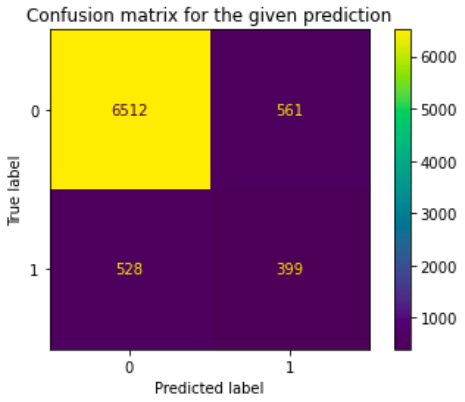
Area under the curve is used to see how the logistic regression model does in classifying the data. AUC for this is 0.838 We know that AUC is better when closer to 1. But, from confusion matrix we can say that logistic regression is not a better fit for this problem

Decision Tree:

We are using tree-based classifiers instead of linear classifiers to improve the accuracy of our model. Decision trees do not require feature scaling as they are not sensitive to variance in data.

```
In [100]: evaluation(y_test,y_pred)
```

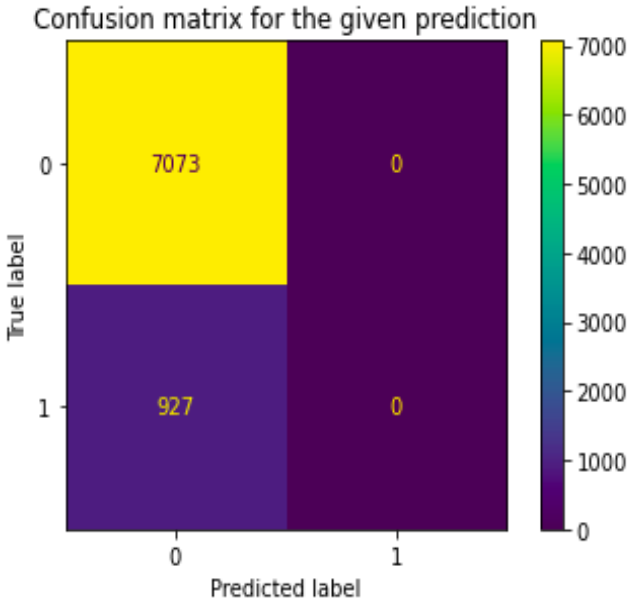
```
Accuracy: 0.863875
f1_score: 0.42289348171701113
Precision: 0.415625
Recall: 0.43042071197411
Classification report:
              precision    recall  f1-score   support

           0       0.93      0.92      0.92      7073
           1       0.42      0.43      0.42       927

    accuracy                           0.86      8000
   macro avg       0.67      0.68      0.67      8000
weighted avg       0.87      0.86      0.86      8000
```



We observe that the accuracy of our model is slightly reduced to 86% using a decision tree. Applying optimization techniques- checking accuracy by adding the criterion gini index.

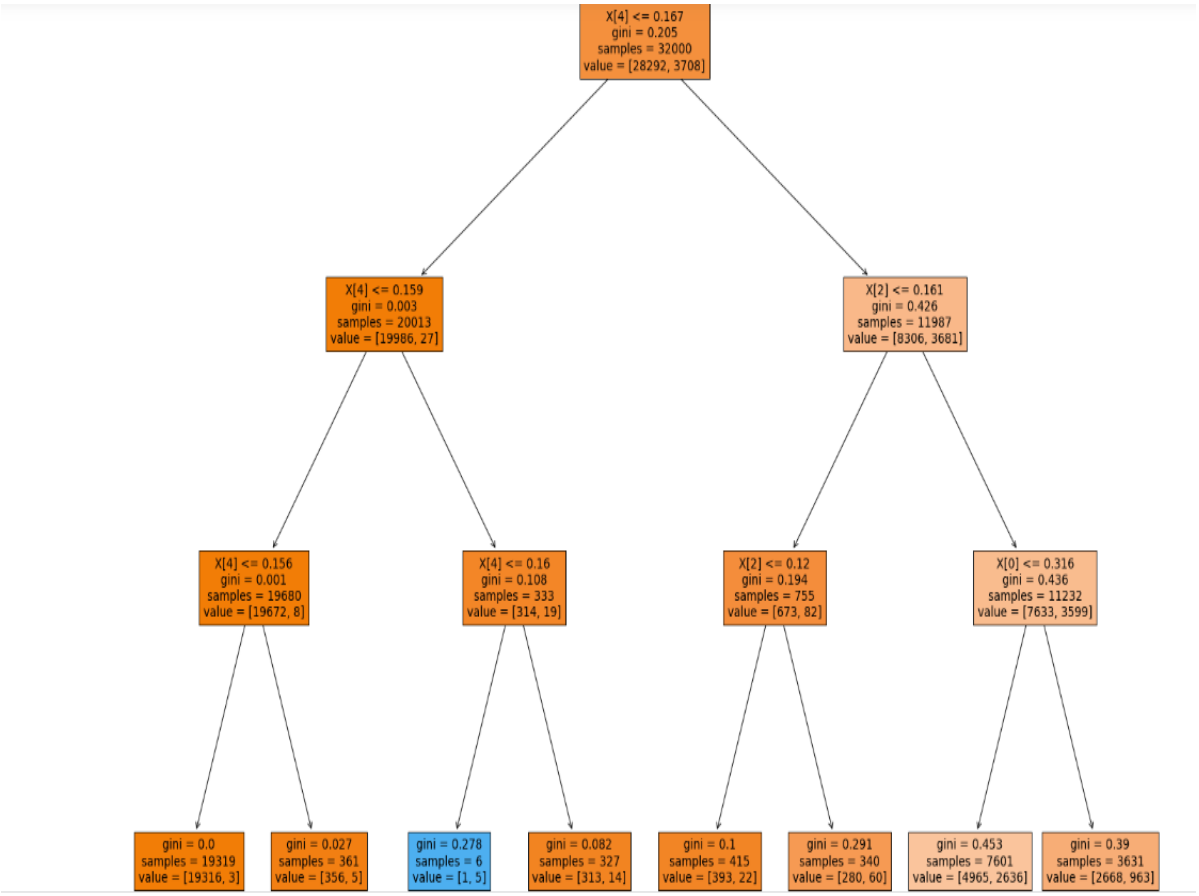
```
In [101]: # instantiate the DecisionTreeClassifier model with criterion gini index

          dt_gini = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)


          # fit the model
          dt_gini.fit(x_train, y_train)

Out[101]: DecisionTreeClassifier(max_depth=3, random_state=0)

In [103]: y_pred_gini = dt_gini.predict(x_test)

In [104]: from sklearn.metrics import accuracy_score

          print('Model accuracy score with criterion gini index: {0:0.4f}'. format(accuracy_score(y_test, y_pred_gini)))

          Model accuracy score with criterion gini index: 0.8841

In [105]: print(classification_report(y_test,y_pred_gini))

                        precision    recall  f1-score   support

                     0       0.88      1.00      0.94      7073
                     1       0.00      0.00      0.00       927

              accuracy                           0.88      8000
             macro avg       0.44      0.50      0.47      8000
          weighted avg       0.78      0.88      0.83      8000
```

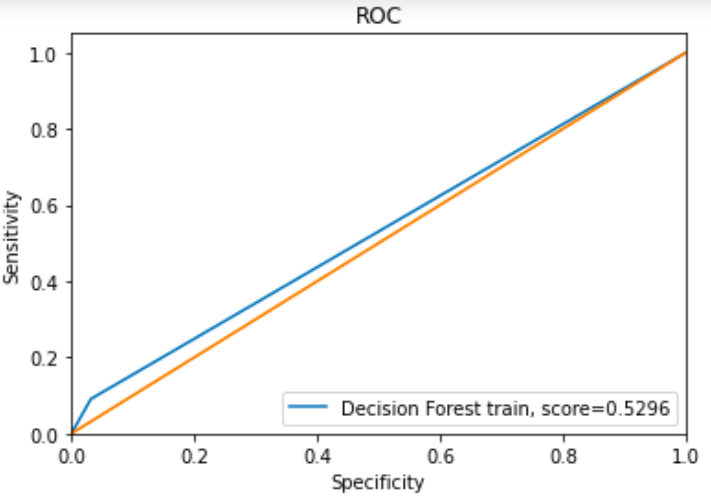
Here, y_pred are true class labels, and y_pred_gini are predicted class labels. The accuracy of the model increased to 88.4% this is a slight increase over the logistic regression model as well. But, this model predicted all 0's even it has 1's in it test data. From this we can say, decision tree does not give the predicted results.

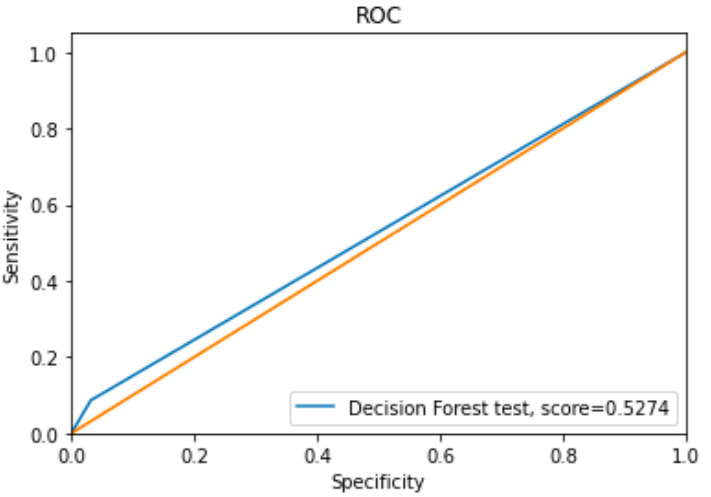


We observe from the confusion matrix that there are no false negatives and true negatives, because of this condition we can say that using decision tree doesnt give proper results in this case. Hence, Decision tree is not the right model to predict results for this dataset.

This is the structure of a single decision tree based on all features together.


X[4] <= 0.167
gini = 0.205
samples = 32000
value = [28292, 3708]
X[4] <= 0.159
gini = 0.003
samples = 20013
value = [19986, 27]
X[2] <= 0.161
gini = 0.426
samples = 11987
value = [8306, 3681]
X[4] <= 0.156
gini = 0.001
samples = 19680
value = [19672, 8]
X[4] <= 0.16
gini = 0.108
samples = 333
value = [314, 19]
X[2] <= 0.12
gini = 0.194
samples = 755
value = [673, 82]
X[0] <= 0.316
gini = 0.436
samples = 11232
value = [7633, 3599]
gini = 0.0
samples = 19319
value = [19316, 3]
gini = 0.027
samples = 361
value = [356, 5]
gini = 0.278
samples = 6
value = [1, 5]
gini = 0.082
samples = 327
value = [313, 14]
gini = 0.1
samples = 415
value = [393, 22]
gini = 0.291
samples = 340
value = [280, 60]
gini = 0.453
samples = 7601
value = [4965, 2636]
gini = 0.39
samples = 3631
value = [2668, 963]





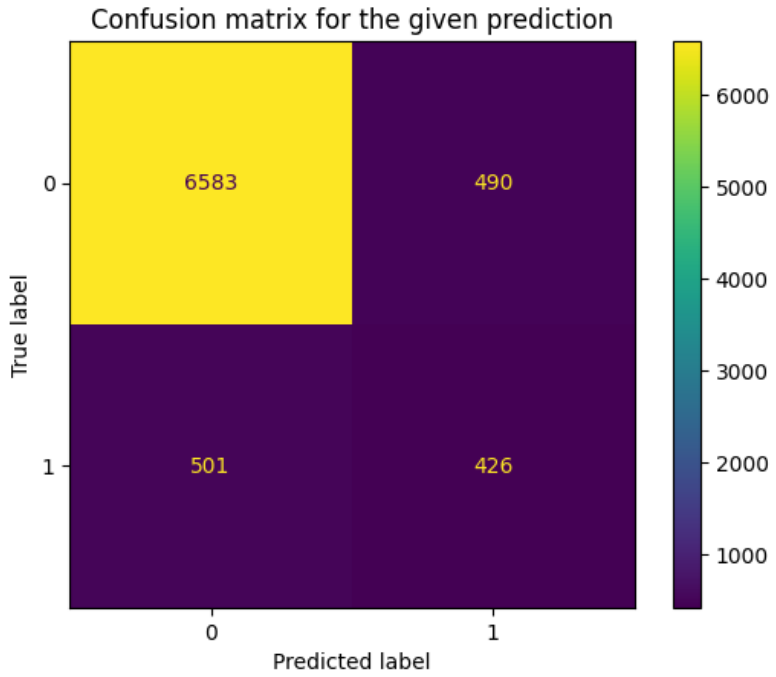
Area under the curve decreased by using a Decision Tree model. Therefore, we understand that the Decision Tree model is poorer in classifying our dataset into hazardous and non hazardous when compared to Logistic regression model. We observe that other metrics like value of recall and accuracy have increased significantly by using a decision tree.

Random Forest Classifier :

While a decision tree combines some decisions, a random forest combines many decision trees. As a result, it is a slow yet lengthy process. A decision tree, especially a linear one, is quick and works well with huge data sets. Thorough training is required for the random forest model. Thats why we have chosen random forest classifier.

With our test and train datasets, we achieved accuracy of 87.6%, precision and recall of 47% and 46% respectively. FP and FN values are indeed very high. In order to get better outcomes, several optimization techniques are included.

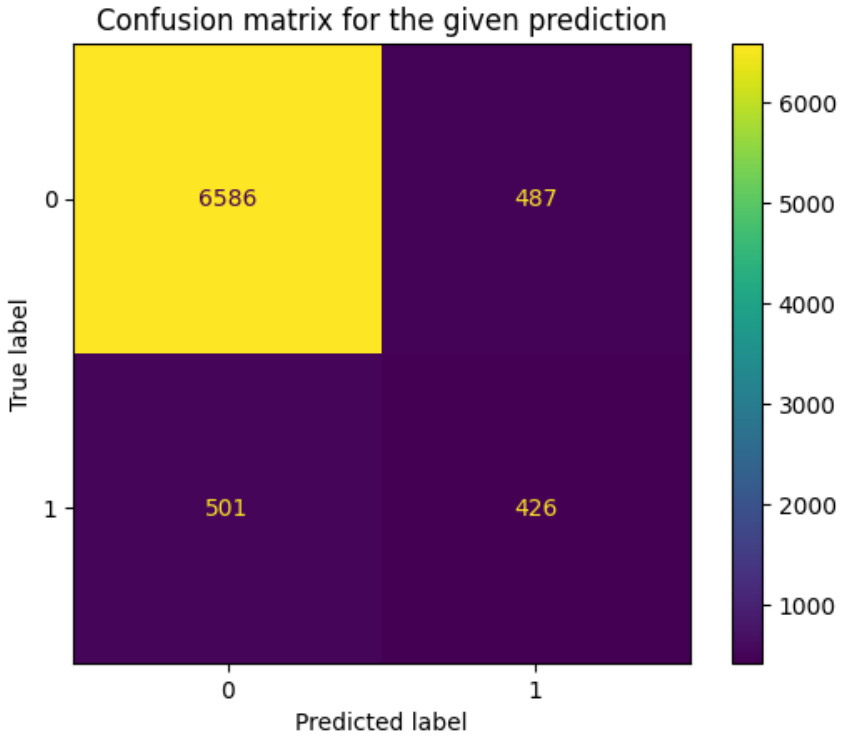
```
Accuracy: 0.876125
f1_score: 0.46228974498100917
Precision: 0.4650655021834061
Recall: 0.459546925566343
Classification report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93      7073
           1       0.47      0.46      0.46       927

    accuracy                           0.88      8000
   macro avg       0.70      0.70      0.70      8000
weighted avg       0.88      0.88      0.88      8000
```



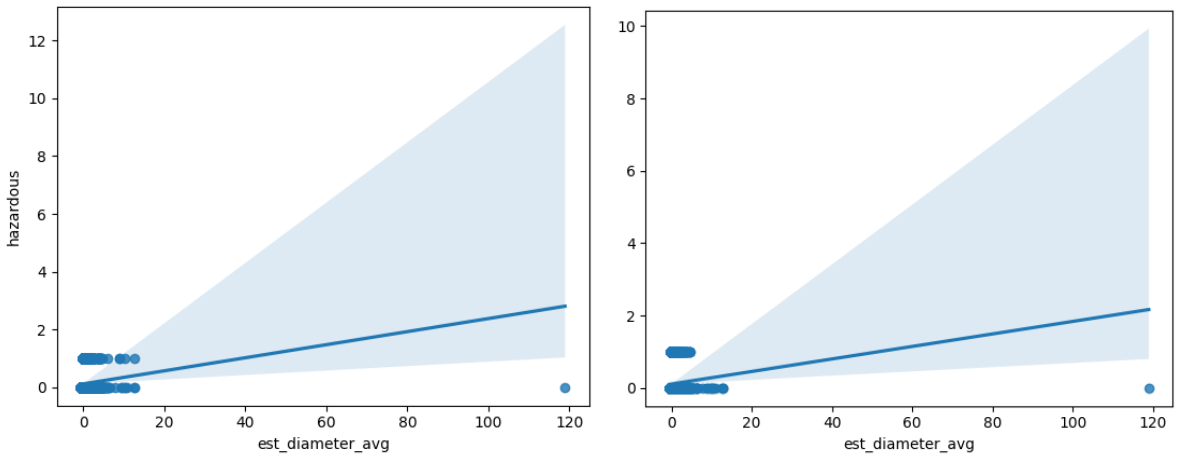
The "Entropy" criterion has been employed as the optimization criterion in this case. The Entropy Index for a feature's decline at each split is a measure of its relevance, and Random Forests give us this ability to examine it. A feature is more significant when its Entropy Index value falls.

```
Accuracy: 0.8765
f1_score: 0.4630434782608695
Precision: 0.4665936473165389
Recall: 0.459546925566343
Classification report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93      7073
           1       0.47      0.46      0.46       927

    accuracy                           0.88      8000
   macro avg       0.70      0.70      0.70      8000
weighted avg       0.88      0.88      0.88      8000
```

We observe slight increase in the accuracy from 87% to 87.6% after using optimization techniques. Also, with the slight increase in the precision.

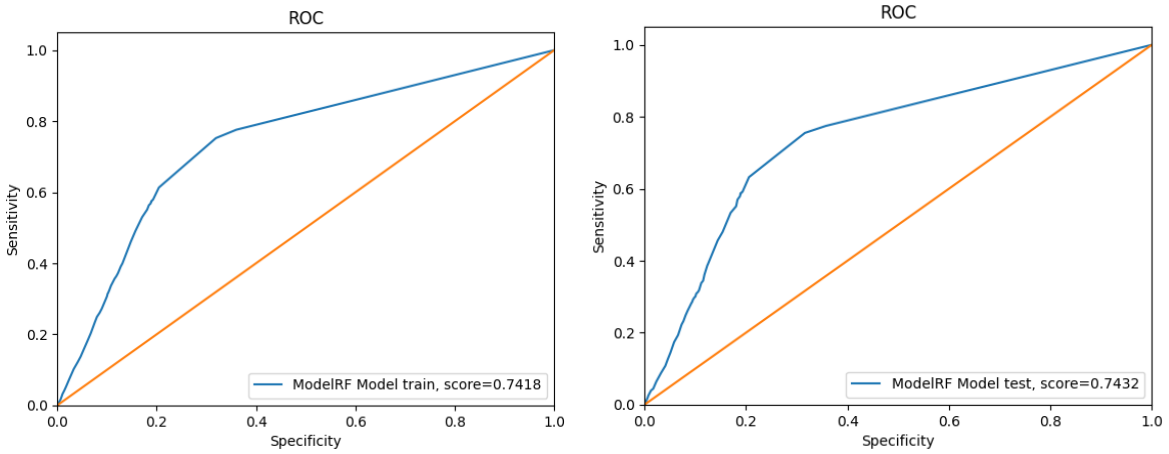


The x test, y test, and predicted y test are all displayed in the graphs below. We can say that the outcomes i.e., both y_test and y_pred gives similar results. In both instances, asteroids are categorized as hazardous when est diameter avg increases.



Receiver operating characteristic - curve :

The ROC curve works best for determining how well a classifier performs. It illustrates the link between specificity and sensitivity. The roc-curve-score is 74.18% and 74.32% for train and test data set respectively. Which means our model is not classifying the feature properly w.r.t other models, as seen in the graph.
Classifiers that provide curves that are nearer to the top-left corner exhibit better performance. Our model is reliable because our curve is oriented more to the top-left.



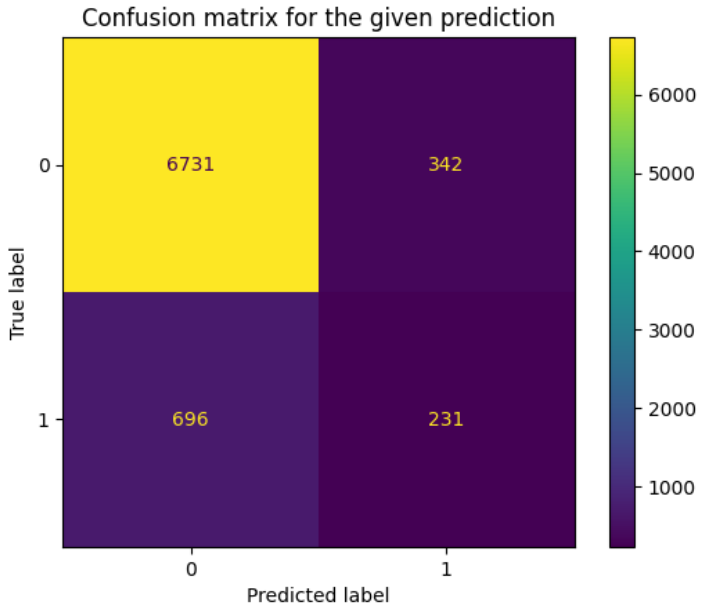
We observe that there is slight decrease in the accuracy rate when to decision tree. It is because, in random forest classifiers we are considering many decisions where as it is not the case in decision tree. Also Decision tree shown best results when compared to random forest classifier.

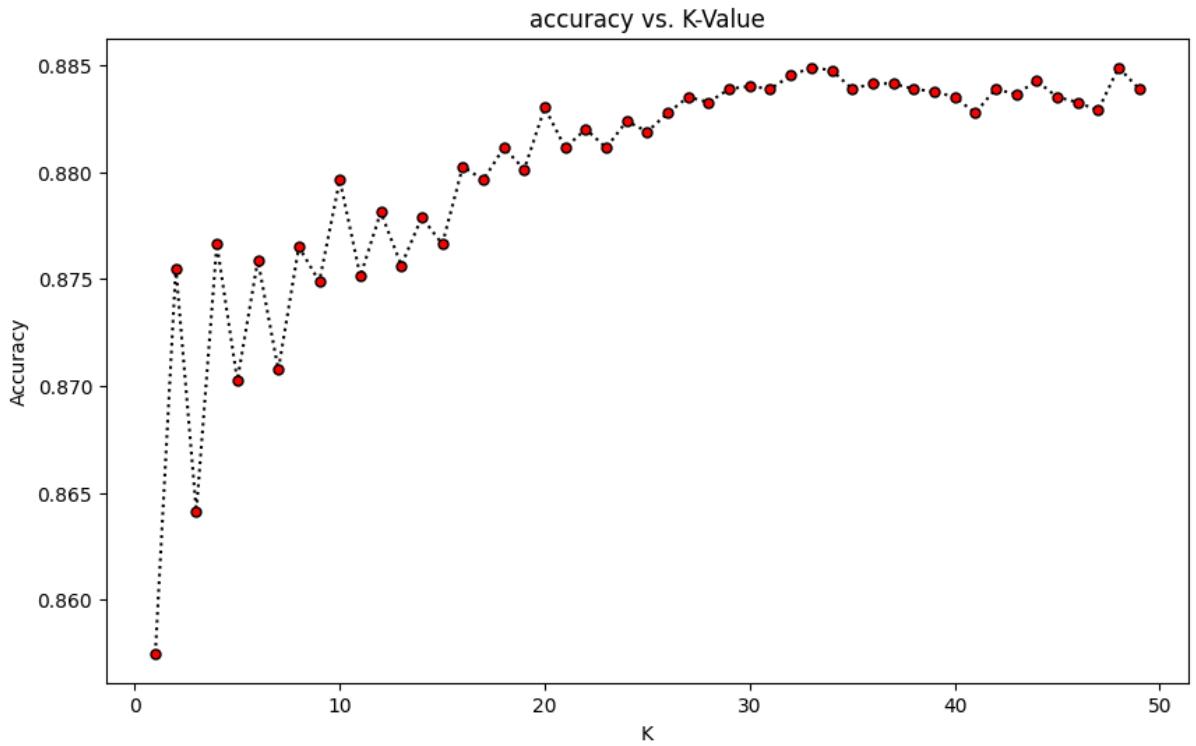
K-Nearest Neighbours:

We chose KNN, as it is a simple machine learning algorithm that performs both classification and regression tasks without any parameters required for assumption. The main intuition is to find the distance between points and select k nearest data points.

```
Accuracy: 0.87025
f1_score: 0.30800000000000005
Precision: 0.4031413612565445
Recall: 0.24919093851132687
Classification report:
              precision    recall  f1-score   support

           0       0.91      0.95      0.93      7073
           1       0.40      0.25      0.31       927

    accuracy                           0.87      8000
   macro avg       0.65      0.60      0.62      8000
weighted avg       0.85      0.87      0.86      8000
```

We observed an accuracy of 87% while using KNN with a base value of k. Moreover, the values of FP and FN are quite high. By applying a few optimization strategies, we can improve the accuracy.

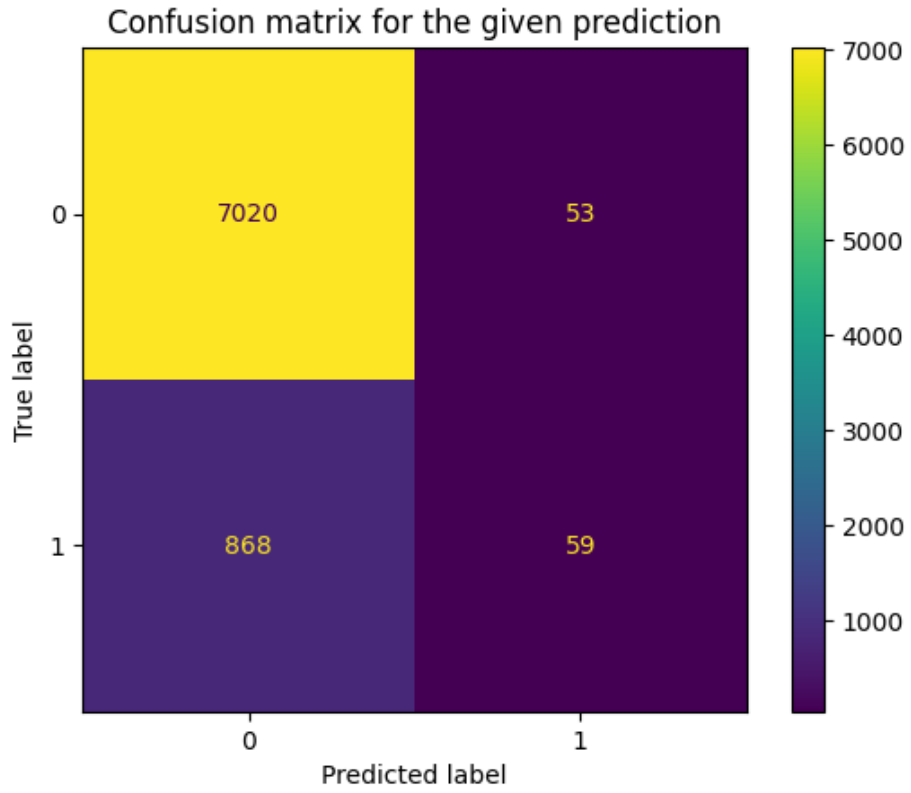




We can see from the graph above that accuracy is greatest when k is 33. As we can see, the accuracy is currently 88.48%. Through the use of the cross-validation technique, the ideal value of k is discovered.

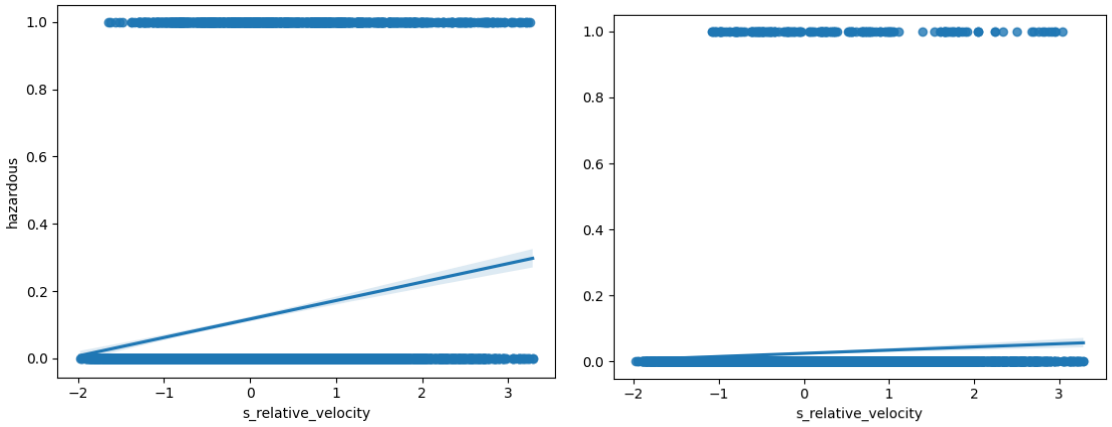
```
Maximum accuracy:- 0.884875 at K = 33
Accuracy: 0.884875
f1_score: 0.11357074109720885
Precision: 0.5267857142857143
Recall: 0.06364617044228695
Classification report:
              precision    recall  f1-score   support

           0       0.89      0.99      0.94      7073
           1       0.53      0.06      0.11       927

    accuracy                           0.88      8000
   macro avg       0.71      0.53      0.53      8000
weighted avg       0.85      0.88      0.84      8000
```

Along with accuracy, there is an improvement in the value of precision from 40% to 52%. This indicates that cross-validation along with KNN improves accuracy for this dataset.

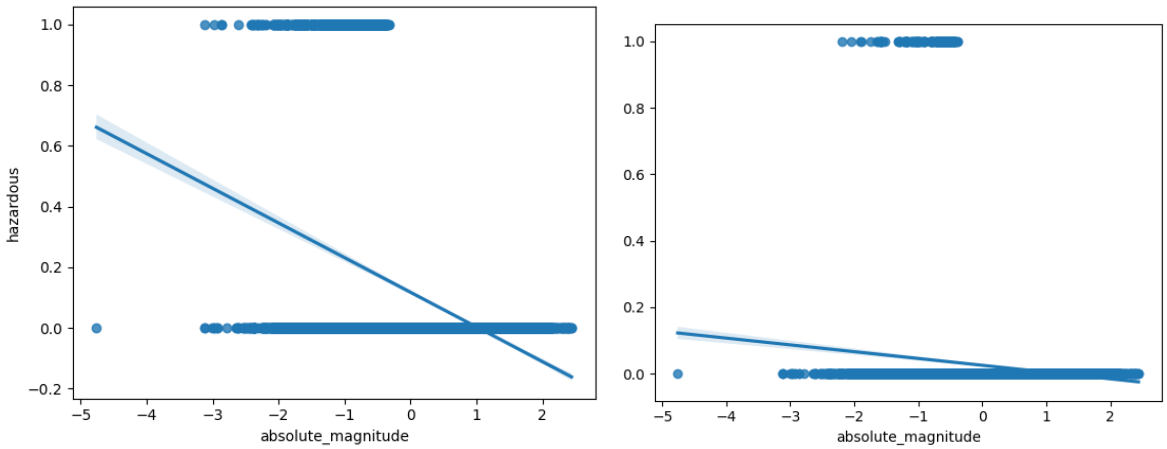


Above is the confusion matrix, a table that allows us to know how well a classification model performs on a set of test data so that the true values may be determined. We can observe that there is a decline in the false positive rate after thoroughly comparing the two confusion matrices, which increases the accuracy of our forecast.

Graphs of x test versus y test and x test versus predicted y test are shown below. With an increase in relative velocity, the asteroids are categorised as hazardous in both scenarios.



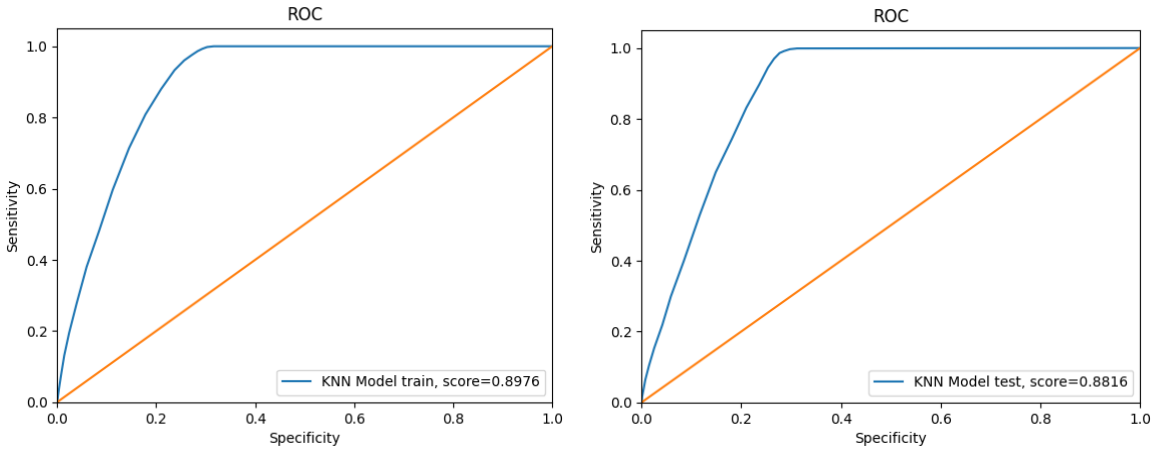
In the graphs below, x test and y test are shown alongside x test and predicted y test. The asteroids are categorised as not hazardous in both scenarios with an increase in absolute magnitude.



Receiver operating characteristic - curve :

Roc curve is best used to identify the performance of the classifier. It draws relationship between sensitivity (true positive rate) and specificity (false positive rate). From the graph, we can say that the roc-curve-score is 55.1%.

A better performance is shown by classifiers that provide curves that are closer to the top-left corner. The test is less accurate the closer the curve gets to the ROC space's 45-degree diagonal. Our curve is towards the top-left, which makes our model reliable.

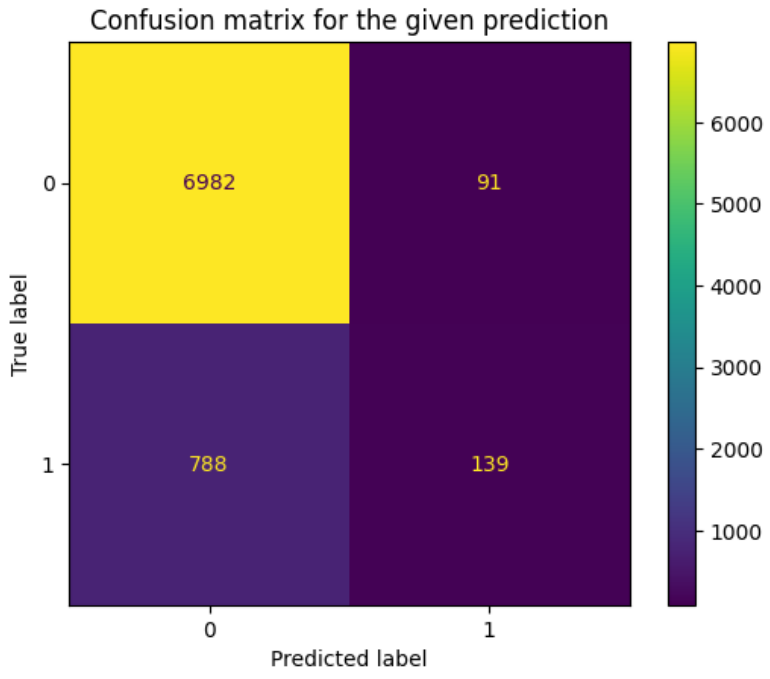


XGBoost :

XGBoost is an ensemble learning method which combines the predictive power of multiple learning models. In XGBoost Decision trees are built sequentially correcting the errors over previous trees. XGBoost implements parallel processing at the node level.

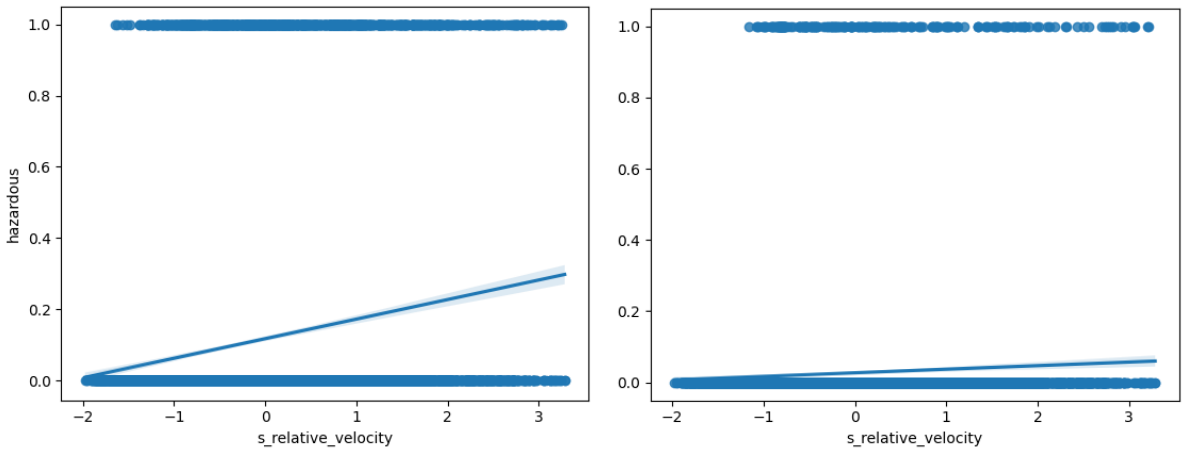
XGBoost model improves the model prediction in many ways, Missing values or data processing steps like one-hot encoding make data sparse. XGBoost incorporates a sparsity-aware split finding algorithm to handle different types of sparsity patterns in the data

```
89.01
Accuracy: 0.890125
f1_score: 0.24027657735522903
Precision: 0.6043478260869565
Recall: 0.1499460625674218
Classification report:
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      7073
           1       0.60      0.15      0.24       927

    accuracy                           0.89      8000
   macro avg       0.75      0.57      0.59      8000
weighted avg       0.86      0.89      0.86      8000
```

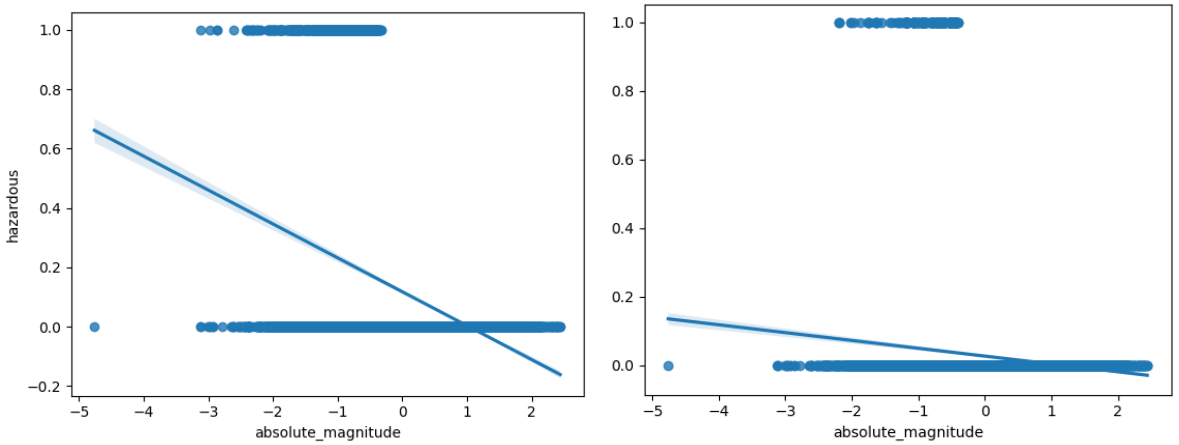


We see that employing the XGBoost model, accuracy climbed to a maximum of 89.01%. We can also notice that the precision rate has grown.

Below are graphs showing the x test versus the y test and the x test versus the projected y test. Both scenarios classify the asteroids as hazardous as they increase in relative velocity.



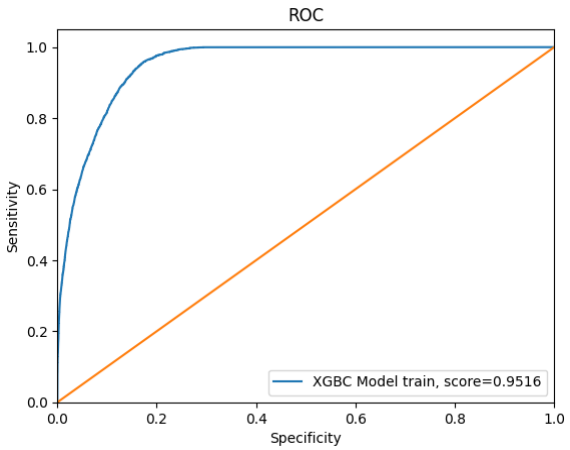
In both cases, the asteroids are categorized as not hazardous despite increasing in absolute magnitude.

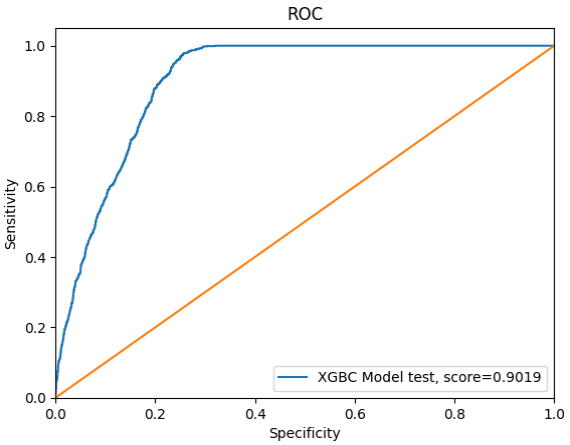


ROC Curve :

We visualized sensitivity and specificity—the two criteria used to assess how well a model fits a dataset—using ROC curves. For each potential decision threshold of an xgboost, pairs of the true positive rate and the false positive rate are plotted on the ROC curve.

Area under the curve is used to see how the xgboost model does in classifying the data. AUC for this is 0.9516 and 0.9019. We know that AUC is better when closer to 1. So, for this model AUC closer to 1.0 means it is good at classifying observations.





Conclusion:

Logistic regression	88.325%
Decision tree	88.41%
Random forest classifier	87.65%
K-Nearest Neighbours	88.48%
XGBoost	89.01%

From above table, The accuracy slightly increasing from model to model. Even after using many optimization techniques we gained accuracy of 88 and 87 for decision tree and random forest. But, XGboost shown incredible performance on both train and test data (we can see this from the roc graphs given). Thus, XGBoost is performing well on our dataset and has high accuracy rate when compared to others, giving best predictions/ results.

We can use XGBoost classifier to determine the likelihood of the incoming object, whether it is hazardous to earth or not.

References:

[1] Decision Tree - https://scikit-learn.org/stable/modules/tree.html

[2] Random forest classifier - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[3] XGBoost - https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier