Classifying Near-Earth Objects

HarvardX 125.9x Capstone Project Part 2 - CYO

Julian Fuchs

October 05, 2024

Contents

1 Introduction **2**

1.1 Motivation . . . 2

1.2 The Dataset . . . 2

2 Analysis **3**

2.1 Cleaning the Data . . . 3

2.2 Exploring the Data . . . 3

2.3 Measuring Accuracy . . . 5

2.4 Train / Test set . . . 5

2.5 A Trivial Approach . . . 6

2.6 Modelbuilding . . . 6

2.6.1 Recursive Partitioning (Rpart) . . . 6

2.6.2 Gradient Boosting (GBM) . . . 8

2.6.3 Random Forest . . . 9

3 Results **10**

4 Conclusion **10**

5 References **11**

1 Introduction

1.1 Motivation

This is the second assignment on the edX HarvardX 125.9x Data Science Capstone course. The goal of this project is to build a machine learning algorithm to classify if an near earth object is hazardous or not. Our aim is to build an algorithm with an accuracy $\geq 95\%$. The Dataset used in this project is provided by Ivansher on Kaggle[1]. A copy of the Dataset is provided on the Github Repository of this project[2].

1.2 The Dataset

Classifying Near-Earth Objects (NEO) sound amazing, but what is a Near-Earth Object? A NEO is described as asteroids or comets, which were formed about 4.6 billion years ago alongside with the planets of our solar system and come near earth on their way threw space[3]. Some of which could be a potential threat to us as an impact could, depending on the object, have catastrophic consequences. We are trying to classify if the objects, included in the dataset, pose a threat or not. The Dataset contains observations from 1910 - 2024.

The dataset includes the predictor as well as 8 Features with 338199 observations:

- **Predictor:** is_hazardous: A logical value indicating if the object is hazardous
- **Features:**
  - **neo_id:** Unique Identifier for each Object
  - **name:** The name of the Object given by NASA
  - **absoulute_magnitude:** Intrinsic luminosity
  - **estimated_diameter_min:** Minimum Estimated Diameter in Kilometres
  - **estimated_diameter_max:** Maximum Estimated Diameter in Kilometres
  - **orbiting_body:** Name of the planet the object is orbiting
  - **relative_velocity:** Velocity Relative to Earth in Kmph
  - **miss_distance:** Distance in Kilometres missed

The following Sections will guide you through the Data cleaning and exploration, how we choose a final model using the Recursive Partitioning, Gradient Boost and Random Forest algorithms to make our predictions and a Conclusion of the Project.

2 Analysis

2.1 Cleaning the Data

Table 1: NA Count

Name	NA Count
neo_id	0
name	0
absolute_magnitude	28
estimated_diameter_min	28
estimated_diameter_max	28
orbiting_body	0
relative_velocity	0
miss_distance	0
is_hazardous	0

As you can see in Table 1 there are some missing values ("NA") in our Data. Most of the algorithms/techniques used can't deal with NA-Values. As the NA/Observations ratio is quite small we can simply remove the rows containing NA's.

The *orbiting_body* feature is the name of the planet the object is orbiting. The problem with this feature is that it has only one unique value: "Earth", as you can see in Table 2. As they are all near earth objects it makes sense but unfortunately it won't be helpful to our analysis. So we can ignore the Feature completely. The same problem applies to the *name* feature as it is a character vector we can't really do anything with it so we will drop it. As our Data is now cleaned we can go one step further into the exploration.

Table 2: Unique Values

Value	Count
Earth	338171

2.2 Exploring the Data

By exploring the data, we can draw first assumptions about the relationships between features and the predictor and to gain a first insight at what algorithm might be useful. A standard linear model $y = ax + b$ would be one of the most basic approaches. A good starting point is to look at the Correlation between the features and the predictor.

Table 3: Correlation

	is_hazardous
neo_id	-0.1567687
absolute_magnitude	-0.3439959
estimated_diameter_min	0.1648414
estimated_diameter_max	0.1648414
relative_velocity	0.1870343
miss_distance	-0.0065423
is_hazardous	1.0000000

As you can see in Table 3 *absolute_magnitude* and *is_hazardous* have the strongest linear relationship. However as -0.3 is not enough to build a good standard linear model, we have to think about other algorithms.

Because there are more than 300k data points per feature in the original Dataset, drawing conclusions would be quite difficult. So, in the following plots we are using a representative 0.5% subset in order to look for relationships in the data.

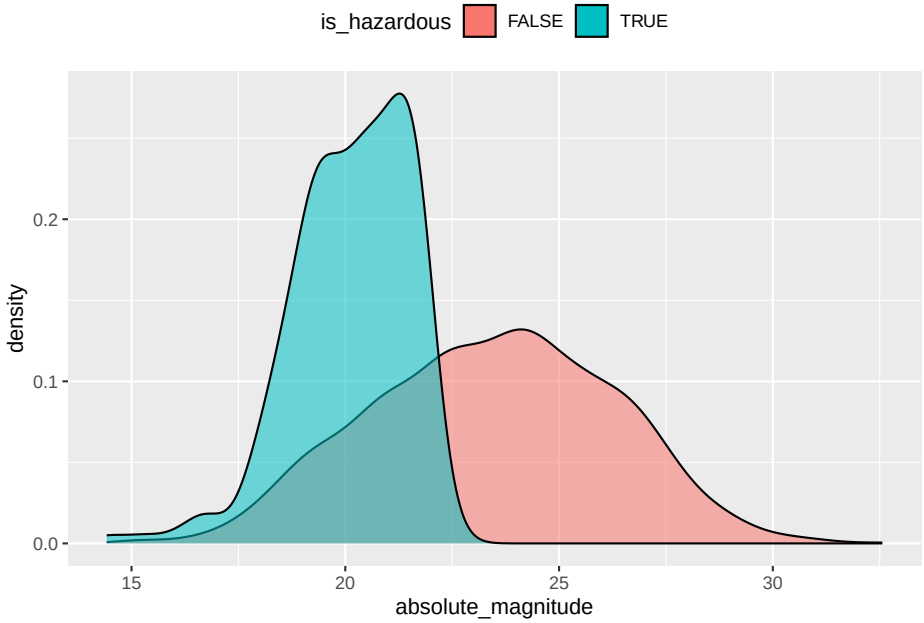

Figure 1: Density

As you can see in the Figure 1, the density of *absolute_magnitude* shows an interesting insight. All hazardous objects have a magnitude ≤ 22.54. This indicates that it could be a valuable feature in our later analysis.

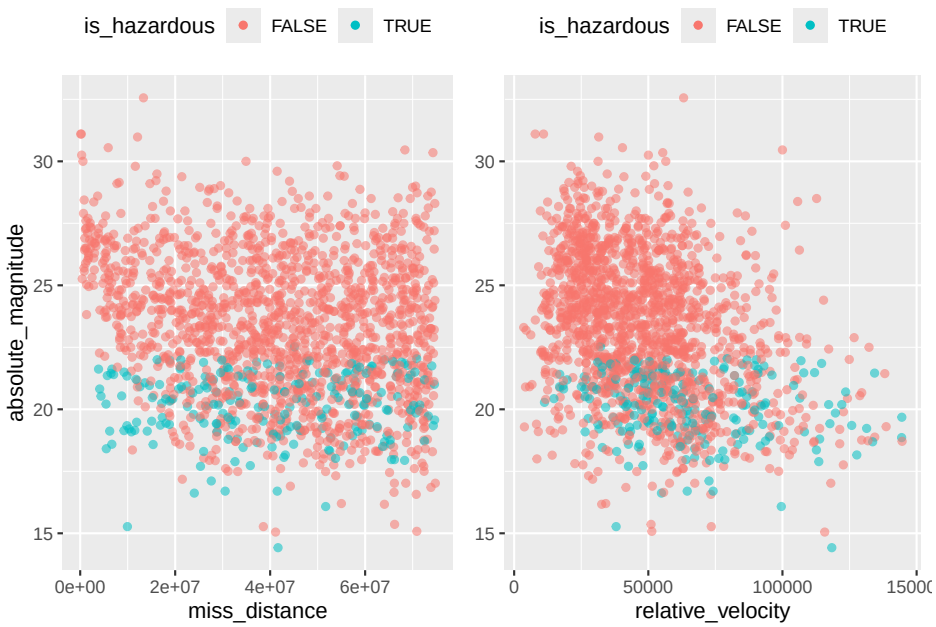

Figure 2: Relationship

As you can see in Figure 2, there is evidence of clustering. This, together with what we found in Figure 1, indicates that decision tree based models, like Rpart, could be quite useful on this data.

2.3 Measuring Accuracy

There are many approaches of measuring error. But because *is_hazardous* is a binary variable, the predicted value can either be identical to the actual value or not. Therefore it makes sense to measure the accuracy in percent.

We will measure the accuracy of our model using the following formula:

$$\textbf{Accuracy} = \frac{1}{n}\sum_{i=1}^{n} f(\hat{y}_i, y_i) \quad \text{where} \quad f(x_1, x_2) = \begin{cases} 1 & x_1 = x_2 \\ 0 & x_1 \neq x_2 \end{cases}$$

In this formula $\hat{y}$ are the actual values and y are the predicted. Translated into code it looks like this:

```
accuracy <- function(y_actual, y_predicted) {
  mean(ifelse(y_actual == y_predicted, 1, 0))
}
```

2.4 Train / Test set

In order to train and test our model we will partition the data into 80% and 20% subsets.

- **Training set** - will be used to train the algorithm *(80%, 270536 observations)*
- **Test set** - will be used to validate the algorithms accuracy *(20%, 67635 observations)*

2.5 A Trivial Approach

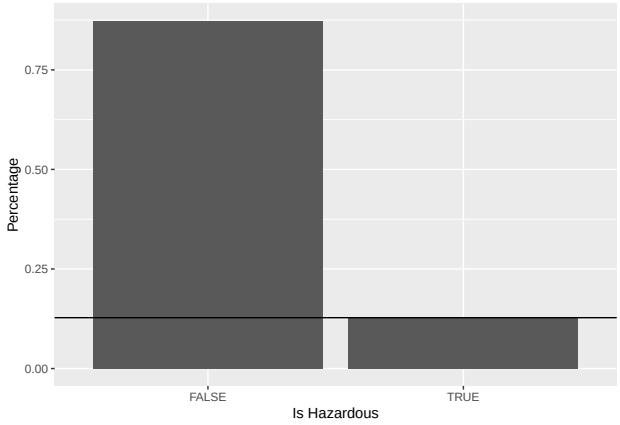

Figure 3: Hazardous Rate

As you can see in Figure 3, most objects are not hazardous. So, the most basic model we can think of is predicting whether a object is hazardous or not with the mean (0.1276318). Because *is_hazardous* is a binary and not a continuous numerical variable we round it to 0.

Model	Accuracy	Goal
Trivial	0.872359	FALSE

This gives us an accuracy of 87.24% straight out of the box. However this isn't quite worth celebrating, because we are basically just predicting all the hazardous objects wrong. You can see it when we add the Accuracy of the model and the Mean of the Data as it will give us approximately 1. So in a real world application, this model could result in catastrophic events.

2.6 Modelbuilding

2.6.1 Recursive Partitioning (Rpart)

Because of the data distribution of some features like *absolute_magnitude* Rpart is a great algorithm to start our model building process.

The Rpart algorithm uses a decision tree, like you can see in Figure 4, to classify the input and predict an outcome.

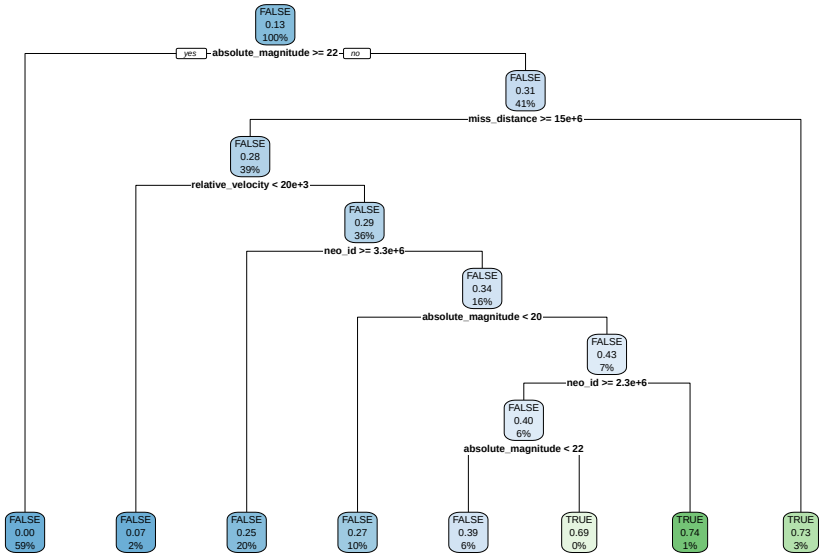

Figure 4: Rpart decision tree

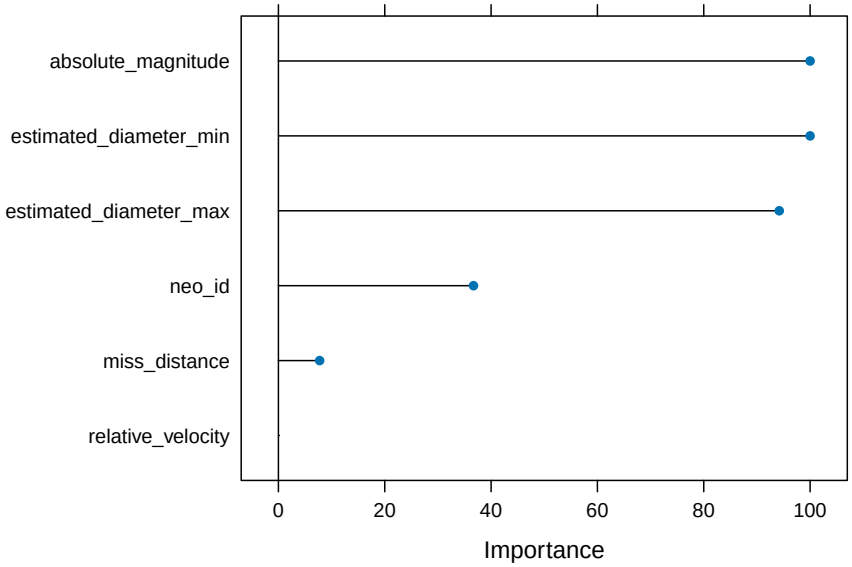

Figure 5: Rpart Feature Importance

You can see in the decision tree (Figure 4) and in Figure 5, on what features the algorithm relies. As indicated before in the data exploration, *absolute_magnitude* has the greatest impact on the model.

Model	Accuracy	Goal
Trivial	0.8723590	FALSE

Model	Accuracy	Goal
Rpart	0.8881053	FALSE

This Rpart model uses just the default parameters and is already achieving an accuracy of ≈ 88.81%. The Rpart algorithm has achieved a slightly better accuracy than the trivial, but not good enough.

2.6.2 Gradient Boosting (GBM)

Gradient Boost relies on decision trees internally. When new trees are being added while training, the gradient decent is calculated in order to minimize the loss while previously added trees keep untouched. [4]

We are using the data subset we were using earlier to visualize the data to find the best fit parameters for the model.

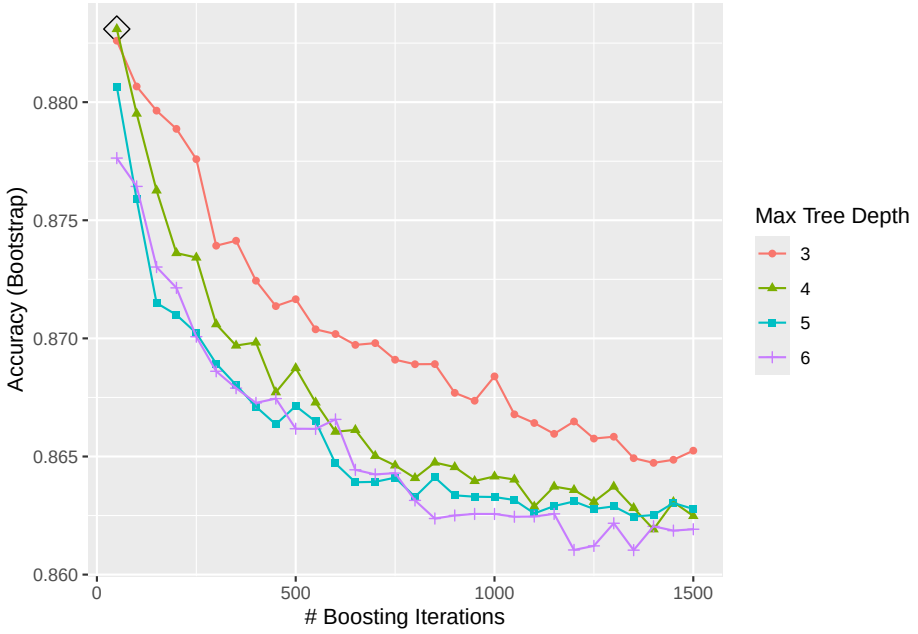

Figure 6: Best tune

You can see in Figure 6 how the models accuracy changes throughout the training and what are best parameters are:

	n.trees	interaction.depth	shrinkage	n.minobsinnode
31	50	4	0.1	10

Using these Parameters we will train the model on the train dataset with crossvalidation to prevent overfitting.

Model	Accuracy	Goal
Trivial	0.8723590	FALSE
Rpart	0.8881053	FALSE

Model	Accuracy	Goal
Gradient Boost	0.8911067	FALSE

We are able to achieve an accuracy of 89.11%. But we can do better.

2.6.3 Random Forest

As our next step we are turning to an algorithm called Random Forest. Random Forest is an ensemble algorithm which consists of multiple low or uncorrelated decision trees and utilizes majority voting to predict the final result. Because of this ensemble of the uncorrelated decision trees it reduces the risk of overfitting the data[5].

As for Gradient Boost, we are using the visualization subset to find the best fit parameters for the model.

The best tune parameter for our Random Forrest algorithm is mtry = 1.

Using these Parameters we will train the model on the train dataset.

Model	Accuracy	Goal
Trivial	0.8723590	FALSE
Rpart	0.8881053	FALSE
Gradient Boost	0.8911067	FALSE
Random Forest	0.9667628	TRUE

We are able to achieve an accuracy of 96.68%.

3 Results

Model	Accuracy	Goal
Trivial	0.8723590	FALSE
Rpart	0.8881053	FALSE
Gradient Boost	0.8911067	FALSE
Random Forest	0.9667628	TRUE

As you can see we reached our goal using our final model, Random Forest, with a accuracy of 96.68% on the test set.

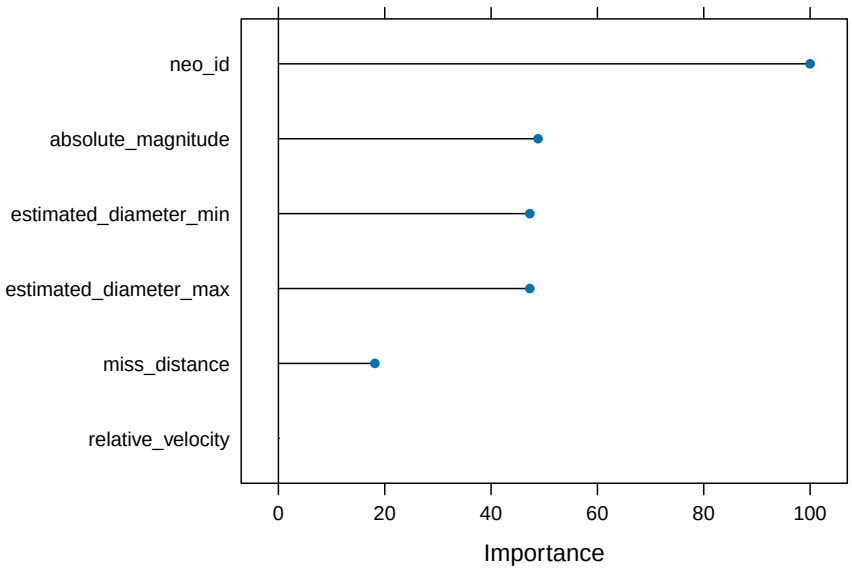

Figure 7: Random Forest Feature Importance

As you can see in Figure 7, The Random Forest algorithm uses *neo_id* as most important feature compared to our Rpart model.

4 Conclusion

In this Project we have tried to predict if a NEO could pose a threat upon Earth based on given parameters. Our goal was to achieve an accuracy $\geq$ 95%. We could accomplish this goal using the Random Forest algorithm reaching an accuracy of 96.68%.

We had 8 features included in the Dataset, however with additional data on the NEO's we could still improve our model further.

5 References

[1] : Ivansher, Kaggle [online] https://www.kaggle.com/datasets/ivansher/nasa-nearest-earth-objects-1910-2024/data [Last Access: Sept 23 2024, Data downloaded: Sept 08 2024]

[2] : https://github.com/Fuechselbruezel/HarvardX-Capstone-CYO

[3]: Nasa Jet Propulsion Laboratory: *NEO Basics*, NASA [online] https://cneos.jpl.nasa.gov/about/basics.html [Last Access: Oct 04 2024]

[4] : Brownlee J.(08/15/2015): *A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning*, Machine Learning Mastery [online] https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/ [Last Access: Oct 04 2024]

[5]: IBM : *What is Random Forest?*, IBM [online] https://www.ibm.com/topics/random-forest [Last Access: Oct 04 2024]