Machine learning for the prediction of local asteroid damage

Grégoire Chomette[1,2]

Lorien Wheeler[1], Donovan Mathias[1]

Asteroid Threat Assessment Project

[1]NASA Ames Research Center

[2]Science and Technology Corporation

Planetary Defense Conference

April 3-7, 2023 – Vienna, Austria

Introduction

- Large **uncertainties** in asteroid properties and trajectories lead to **high numbers** of potential scenarios to adequately cover the parameter ranges
- Planetary defense teams need **fast-running tools** to simulate each situation and evaluate damage probabilities
- Among the solutions proposed, the **PAIR model** (Mathias 2017) simulates **tens-of-millions** of scenarios, with damages and number of people affected
- These studies can be run in **O(1h)** on large **supercomputers**, but they would require **days** on **regular laptops**

→ We propose here to develop **machine learning** models that estimate **accurately** asteroid damages for **millions** of scenarios in **minutes** on regular computers.

Data generation with the PAIR model

The **PAIR** model (Mathias 2017) is used to generate the **dataset** for the ML models. The approach is based on:

- A Monte-Carlo framework to sample realizations from realistic distributions (Fig.1)
- Models to simulate physical mechanisms such as fragmentation, hazard propagation, etc.

We obtain a dataset of **large numbers of data points**, each of them containing:

- A list of 8 parameters characterizing the entry conditions: diameter, velocity, density, incidence angle, aerodynamic strength, luminous efficiency, ablation coefficient, strength scaling coefficient
- The resulting radius of a damaged area (e.g., radius of the >1 psi circular area, Fig.2)

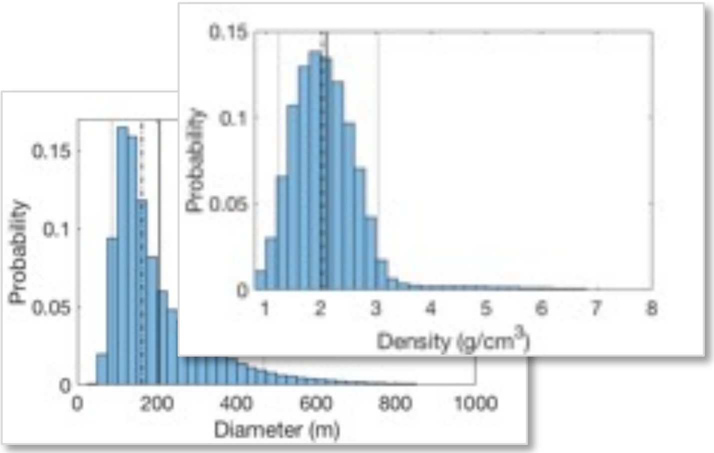


Fig 1: Entry parameter distributions

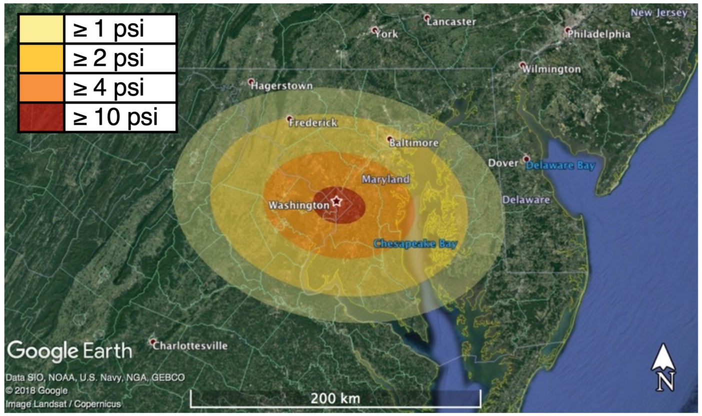


Fig 2: Areas damaged by blast overpressure

Machine learning models

- We are trying to find a function that maps the entry conditions to the associated local damages. In mathematical terms, we try to find a function f such that:

$$f(X) = y \qquad \text{with} \begin{cases} X = [D, v, \theta, \rho, \ldots] & \text{(8 inputs)} \\ y = R_{damaged\ area} & \text{(1 output)} \end{cases}$$

- We propose to train, test and compare 5 machine learning models, in the following order of complexity:
  1. Linear regression
  2. Decision tree
  3. Random forest
  4. Gradient boosting
  5. Neural network

→ The models are trained to adjust their parameters and reduce the sum of squared errors between the predictions and the PAIR output:

$$\text{minimize } \sum_{i=1}^{N_{train}} (y_{i,pred} - y_{i,true})^2 = \sum_{i=1}^{N_{train}} (f(X_i) - y_{i,true})_2$$

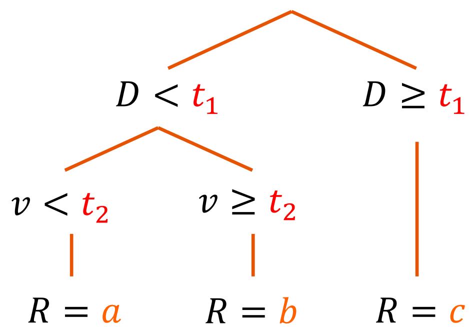
Machine learning models

1. **Linear regression:** a **linear combination** of the independent variables

$$R_{damaged} = f(D, v, \ldots, \theta) = \beta_0 + \beta_1 D + \beta_2 v + \ldots + \beta_n \theta$$

with coefficients $\{\beta_0; \beta_2; \ldots; \beta_n\}$ to be optimized

2. **Decision tree:** a sequence of **comparisons** between independent variables and **thresholds** to determine the value of the prediction at the leaf node

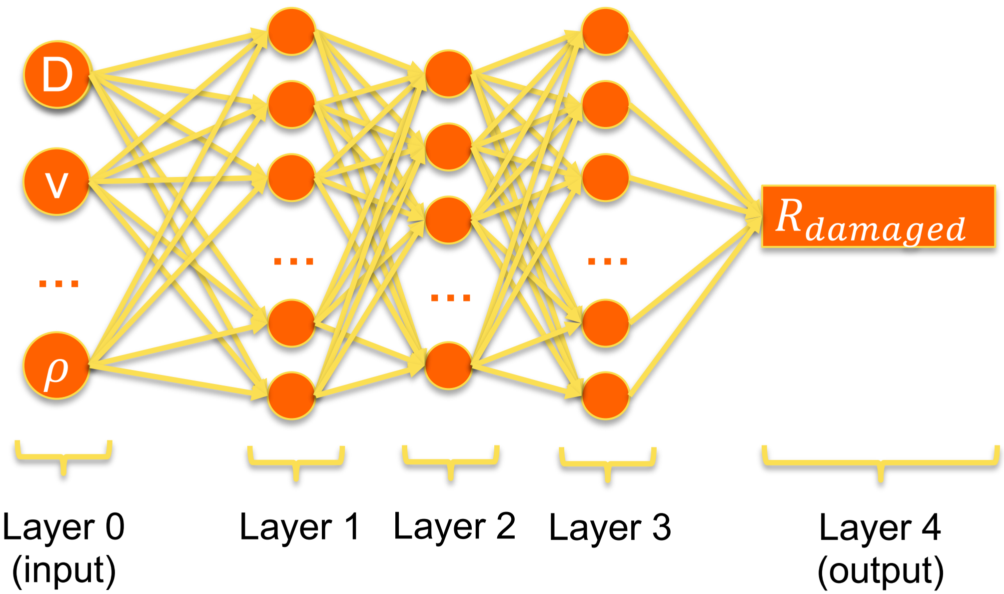


3. **Random forest:** the average of N decision trees trained **independently** on different data subsets. If R_i is the prediction of decision tree i, $R = \frac{1}{N}\sum_{i=1}^{N} R_i$

4. **Gradient boosting:** the weighted combination of N decision trees, where trees are trained **successively** on the residuals of the previous ones to adjust the errors. If R_i is the prediction of decision tree i, $R = \frac{1}{N}\sum_{i=1}^{N} w_i R_i$

Machine learning models

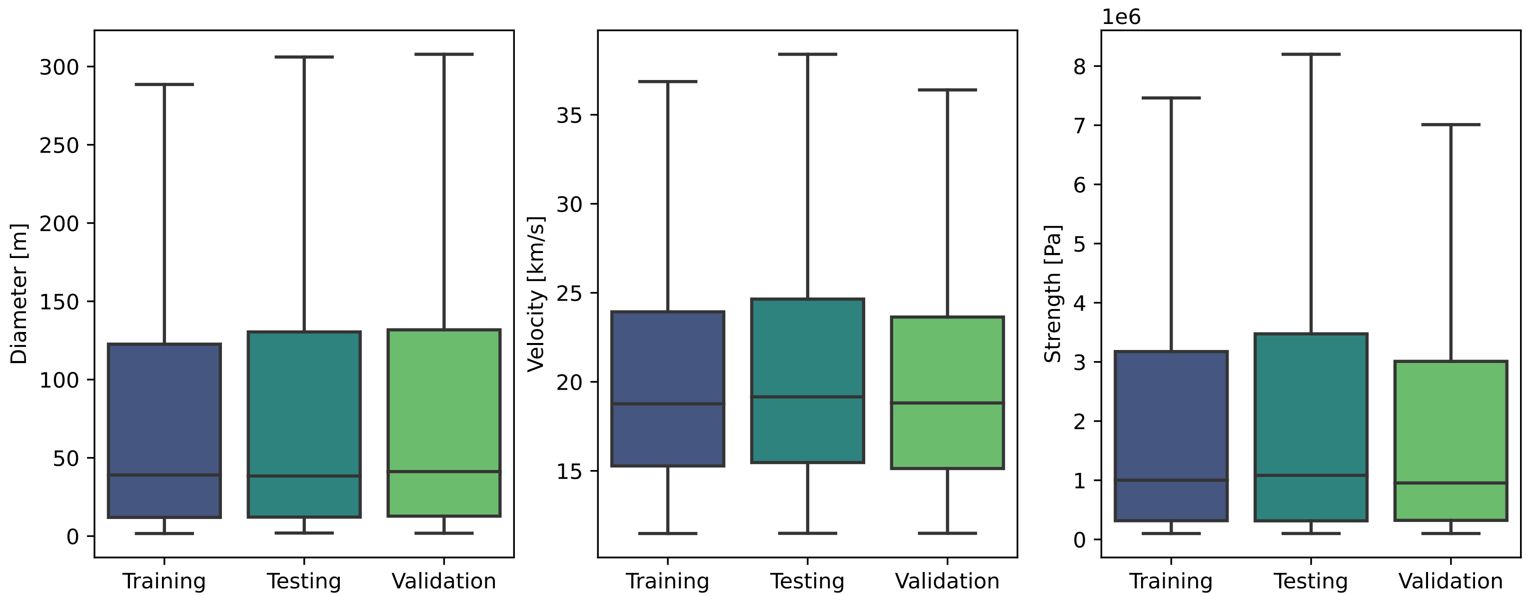
5. **Neural network:** a **complex parametric function** that transforms the input vector several times through L successive layers:



- Each layer consists in a matrix multiplication operation with matrix W^l for layer l, and a transformation with a predefined activation function f^l
- The optimizer of the neural network tries to find the best set of weights in matrices W^l for each layer l in $[1; L]$
- We try several architectures with different numbers of layers, activation functions, etc.

Training, validation and testing

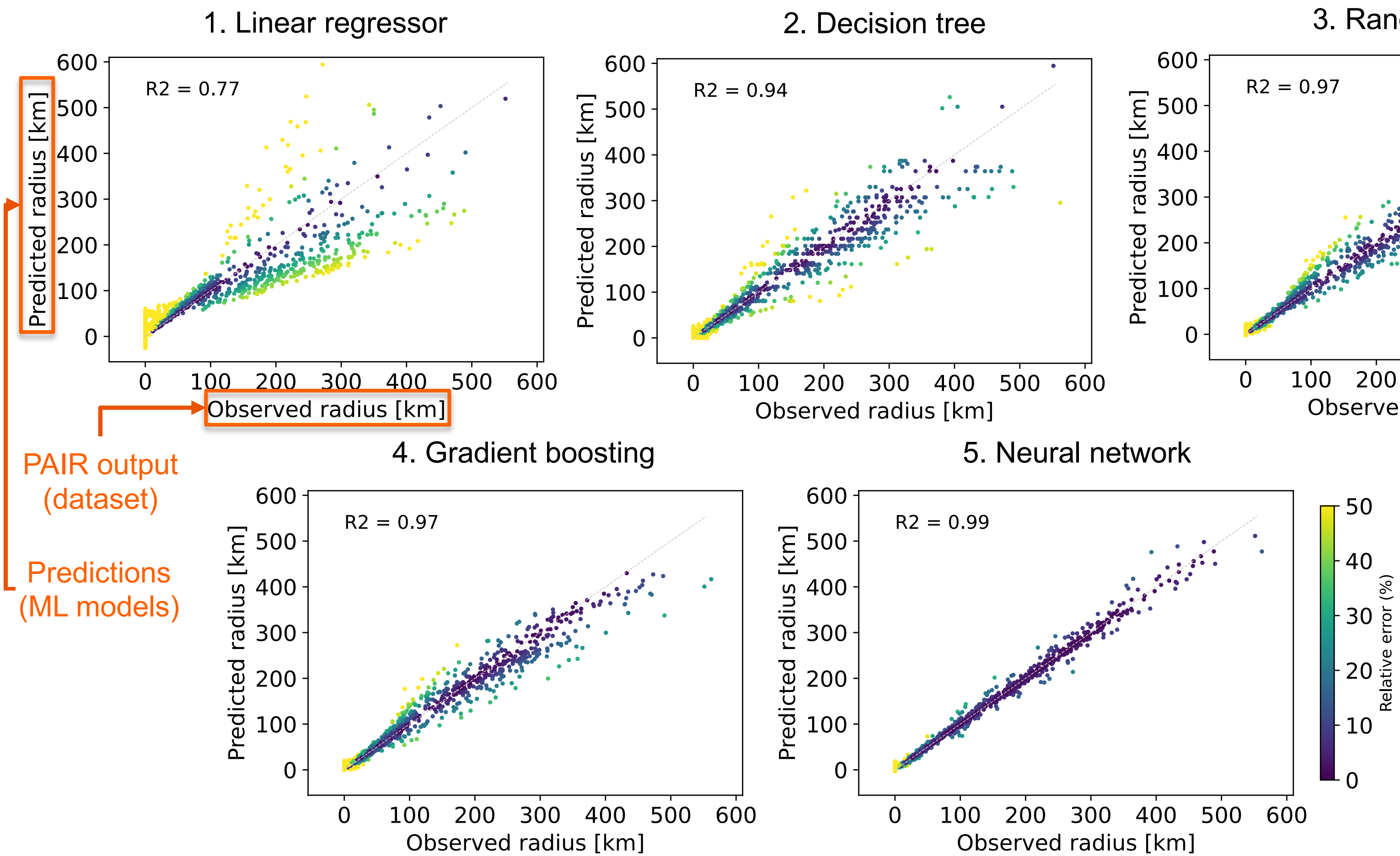
- The original dataset is split into 3 subsets: the training, validation, and test sets with 7000, 2000, and 1000 points respectively. Each subset covers most of the range of entry conditions from the distributions used in PAIR:



- The training set fits the weights of the ML models, the validation set tunes the hyper-parameters, and the test set evaluates the performance on unseen data.
- We use common ML best practices: data normalization, cross validation, etc.

Results: prediction of blast radius

The five machine learning models are trained to estimate the radius of the area damaged by serious blast (i.e., $> 1psi$ blast overpressure):



NN results:

$R2 = 0.99$

$\bar{e}_{abs} = 5.0\ km$

$\bar{e}_{rel} = 11.5\ \%$

Conclusion

We have identified an opportunity to complement physics-based models with ML methods for asteroid risk assessment. The results of our ML models are:

- **Accurate predictions** of the size of damaged areas with ~10% average error on the radius compared to the PAIR model. Coefficients of determination are around 99%, and absolute errors are on the order of a few kilometers

- Significant **reduction** of **hardware requirements**, local asteroid damages can be computed in minutes on local laptops instead of supercomputers

- **Easy integration** for mitigation teams, possibility to differentiate the models to optimize the response

- Complex **sensitivity** analyses to explain the predictions of the models, and determine which parameters are most responsible for the damage