Three-Body Orbital Dynamics Predictor

Project Reflection

Motivation

I have always had a love for space, I find it very fascinating. This project was a fun way for me to relate it to my other interests in data science and ML.

Approach

To study the behavior of interacting three-body gravitational systems, I built a numerical simulation pipeline based on Newtonian gravity in three dimensions. Each system begins randomly and data for these simulations is collected in a csv file. Outcomes were labeled based on predefined physical criteria. These labeled simulations are then used to train and evaluate machine learning models. I also was able to collect more robust data for individual simulations, which produced very insightful data visualizations.

Simulation

The motion of the bodies was governed by Newton's law of gravity in three dimensions. The equations of motion were solved numerically using an adaptive ODE solver. A small softening factor was included to prevent numerical instability during very close encounters.

Each system was simulated for up to **80 years**, unless a physical event occurred earlier.

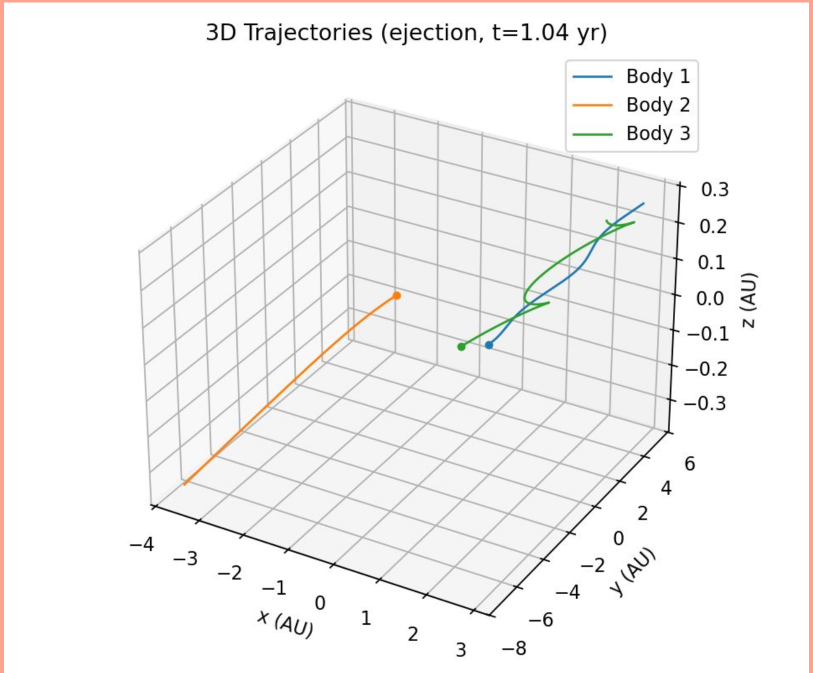
Event Detection

Two types of events were monitored during each simulation:

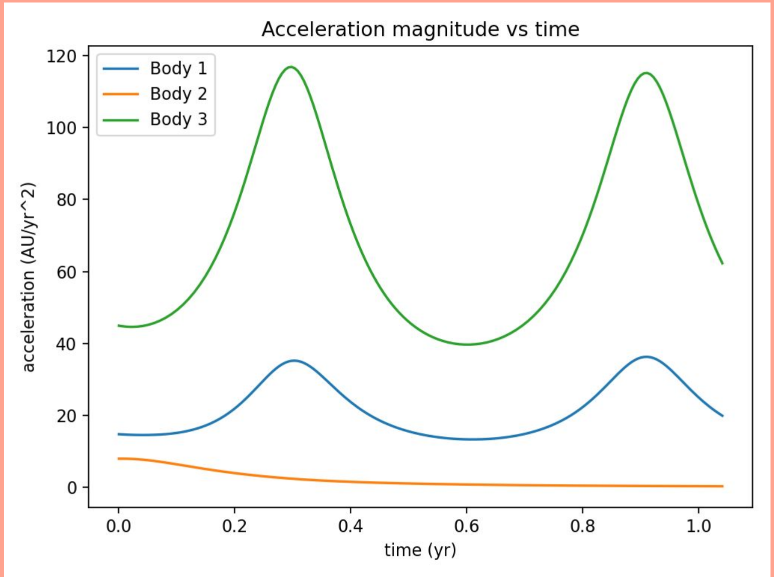
- **Collision:** when any two bodies came closer than a fixed distance threshold.
- **Ejection:** when any body moved farther than a fixed distance from the system's center of mass.

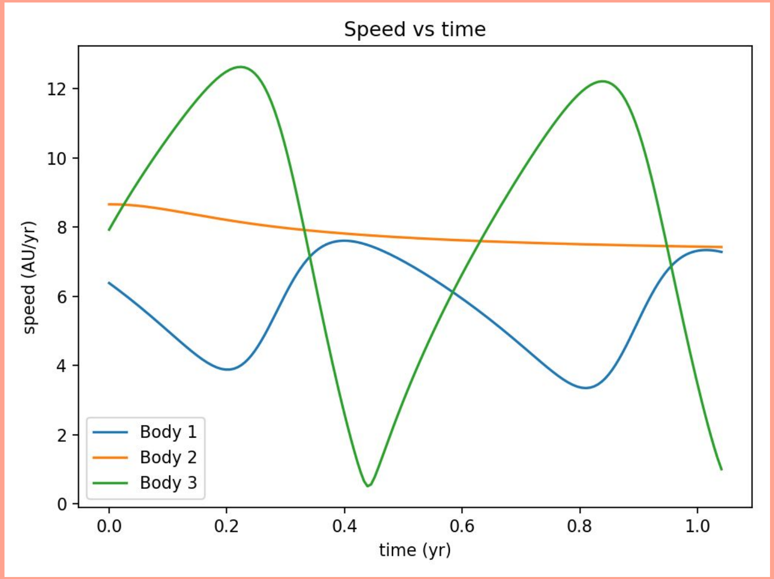
If neither event occurred within 80 years, the system was labeled as **stable**, meaning it stayed bounded for the full time window.

System One

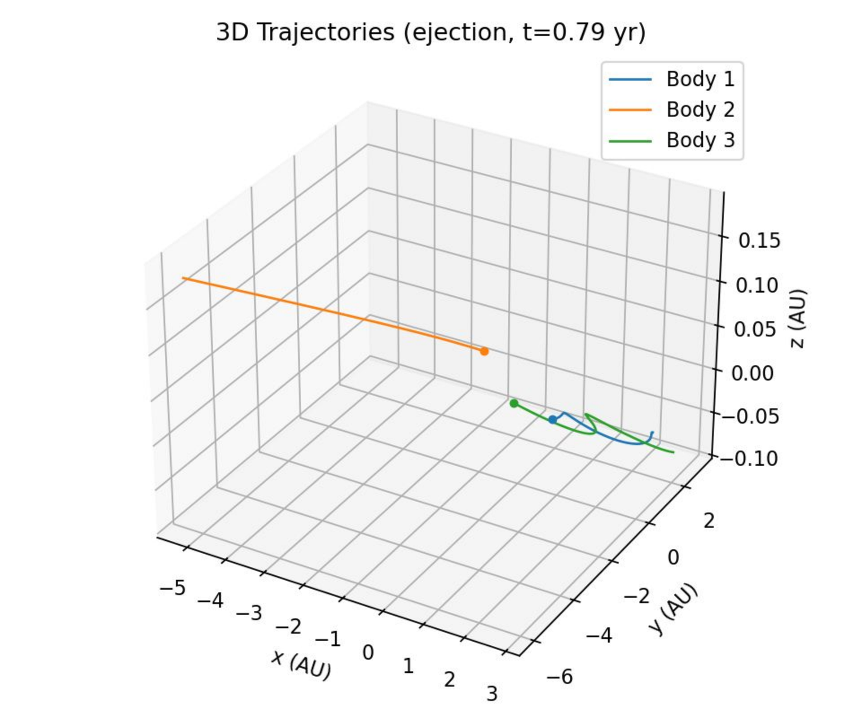

3D Trajectories (ejection, t=1.04 yr)
Body 1
Body 2
Body 3
x (AU)
y (AU)
z (AU)
−4
−3
−2
−1
0
1
2
3
−8
−6
−4
−2
0
2
4
6
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3

System One

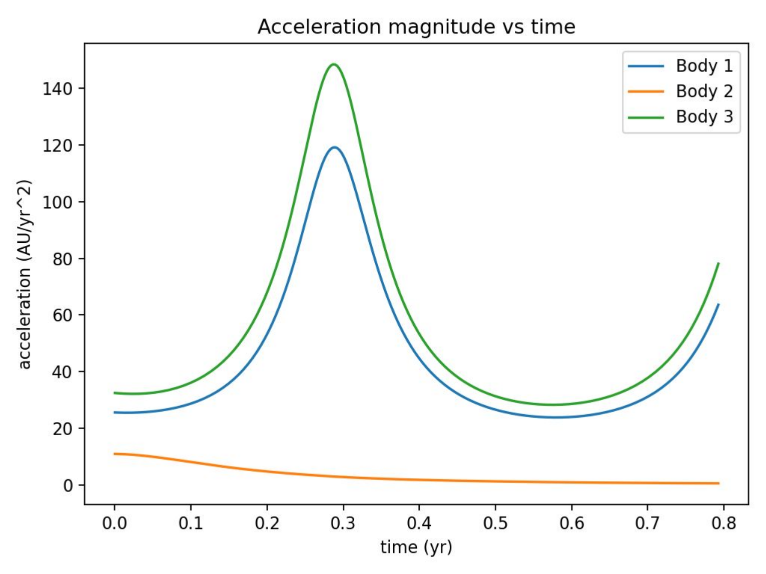

Acceleration magnitude vs time
Body 1
Body 2
Body 3
acceleration (AU/yr^2)
time (yr)
0
20
40
60
80
100
120
0.0
0.2
0.4
0.6
0.8
1.0

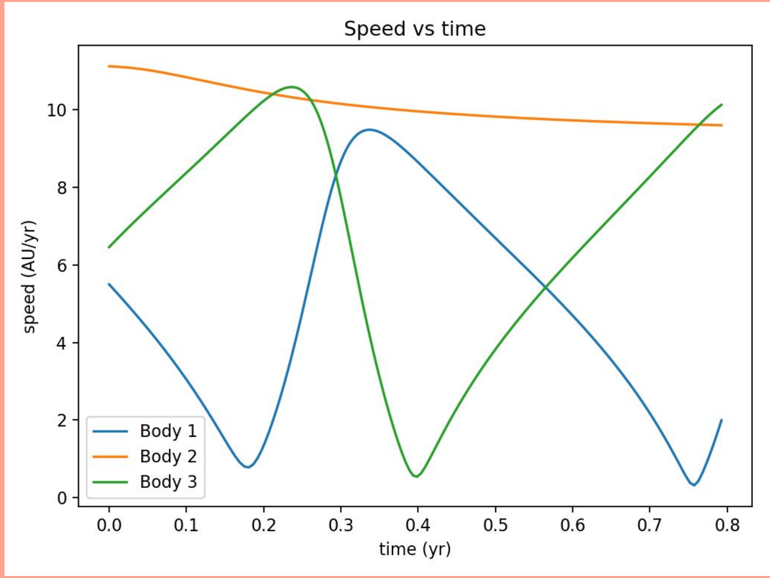

Speed vs time
Body 1
Body 2
Body 3
speed (AU/yr)
time (yr)
0
2
4
6
8
10
12
0.0
0.2
0.4
0.6
0.8
1.0

System Two


3D Trajectories (ejection, t=0.79 yr)
Body 1
Body 2
Body 3
z (AU)
0.15
0.10
0.05
0.00
−0.05
−0.10
x (AU)
−5
−4
−3
−2
−1
0
1
2
3
y (AU)
2
0
−2
−4
−6

System Two

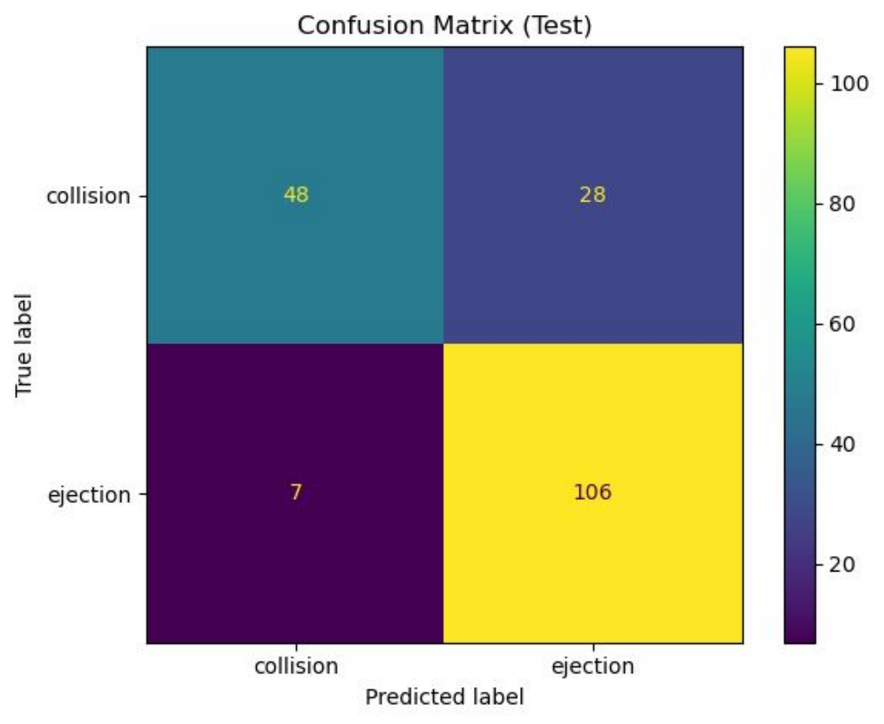

Acceleration magnitude vs time
acceleration (AU/yr^2)
time (yr)
Body 1
Body 2
Body 3
0
20
40
60
80
100
120
140
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


Speed vs time
speed (AU/yr)
time (yr)
Body 1
Body 2
Body 3
0
2
4
6
8
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

Model Evaluation

To predict whether the system results in a collision or ejection, I Trained a fast SVM model (scaled + randomized search). After finding the best parameters I found the accuracy scores for the model.

With an accuracy of 81%, this model is able to consistently predict the final state of these systems. Although the accuracy of collisions is much worse than that of ejections.



Technical Summary

I implemented the simulation and modeling pipeline in Python using NumPy, SciPy, and scikit-learn. Each three-body system is represented as an 18-dimensional state vector containing positions and velocities, and its time evolution is computed by numerically integrating Newton's equations of motion with solve_ivp. I used event functions to terminate simulations when a collision or ejection occurred, allowing outcomes to be labeled efficiently. Initial conditions are generated programmatically with random masses, positions, and velocities, then transformed into the center-of-mass frame to ensure physical consistency. The resulting simulation data are stored in a structured dataset and used to train a Support Vector Machine classifier, with feature scaling handled through a pipeline to ensure stable and consistent model behavior.

Reflection

This project helped me better understand how to build an end to end data science pipeline using simulated data. I generated my own labeled dataset, engineered features, and trained a classification model while paying close attention to preprocessing, class imbalance, and data leakage. Through experimentation and evaluation, I learned that model performance depends not just on the choice of algorithm, but heavily on dataset design and validation, especially when working with simulated data. Overall, this project was a fun exploration of data science applied to physics and simulation-based modeling.