MLFall24

Team 47 - ML Midterm Report

Introduction

For our project, we will be focusing on asteroids, specifically attempting to classify near-earth objects as potentially hazardous, commonly called “asteroid hazard prediction” [3]. Attempts to identify asteroids as hazardous are common, especially using supervised machine learning techniques. It is the weighting of the tracked parameters that differs though, as some focus more on orbiting patterns of asteroids [1] while others prioritize speed and diameter in their supervised models [2], and some even predict impact effects [3]. We plan to focus on a historical NASA dataset, cataloguing near-Earth asteroid information from 1910-2024 and containing features pertaining to general orbital information, diameter measurements, relative velocity, distance missed and a Boolean for whether the asteroid was hazardous or not (our target variable).

Problem Definition

The motivation for our project stems from the fact there are potentially hazardous asteroids in the universe, but because data on them is collected with all other asteroids, sifting through this vast dataset to determine which of these NEOs is hazardous and which isn’t is a time-consuming task that can be made much more efficient. We will leverage the power of machine learning by analyzing this NASA data to help us predict which of these NEOs is hazardous and develop a model which can help predict which of the many further NEOs, which are discovered every day, are hazardous.

Methods

Data Preprocessing Methods

On the preprocessing side, some methods we will use include sampling data, feature engineering, and dimensionality reduction. First, for sampling, the dataset contains over 300,000 entries, so we plan to sample 30,000 to train the models as to be more efficient. For feature engineering, the dataset contains some parameters that describe maximum and minimums of a metric (i.e. diameter), and we can simplify this to an average for some models. Finally, with dimensionality reduction, some of the dimensions of the dataset are unnecessary (i.e. asteroid name) for training the models, so we can remove these to further simplify the dataset.

ML Methods Summary

Because we are using a large dataset, one with an imbalance since only around 13% of the asteroids in the dataset are hazardous, and because we expect several outliers of especially dangerous asteroids, Gradient Boosting is an obvious first pick for a machine learning algorithm. Naive Bayes would be good as well for the same reason of a large sample and because many of our features are conditionally independent (luminosity doesn’t depend on speed or diameter, but can be affected by the miss distance). It can provide a good interpretation if we handle the issues of imbalance in our data set (can be solved by gradient boosting) and the fact it isn’t outlier-resilient [can be mitigated with smoothing]. We also consider Naive Bayes for its speed. Finally, the third model to be used is Logistic Regression, as it is a supervised linear model that is good for basic binary classification when working with large datasets and identifying strong features in correlation to the expected target outputs.

Naive Bayes

The first model we fully implemented was Naive Bayes. Our implementation, which can be found in the NaiveBayes.ipynb file located in this GitHub repository, first began with the reading of our dataset (asteroids.csv in the zip file). With the created Pandas Dataframe, we then dropped rows containing NULL/NA values, as we found there to be 28 rows present with no values in the magnitude and diameter columns. With that done, we then moved the target values (the ”is hazardous” Boolean) into a target array, and did some feature reduction on the dataset, removing the features of “name” and “orbiting body”, as neither were relevant for the model (orbiting body was consistently only “Earth”). We kept “neo_id” just as an entry identifier but removed it when we put the training data in a model. After this, we went ahead and randomly split the data and target values between training and testing (an 80/20 split), and then ran our training data and labels through a gaussian naive bayes function and a logistic regression classifier. We then used the GNB model to predict our testing data’s targets and compared that with our test’s actual target values. From here, we also created a confusion matrix, calculated some basic scoring metrics, and applied 5-fold cross-validation, all of which will be discussed in the Results and Discussion portion.

Logistic Regression

The second model we implemented was Logistic Regression. The implementation can be found in the LogRegression.ipynb file in this Repository. Similar to Naive Bayes, we started by reading the dataset from asteroids.csv, and creating a Pandas Dataframe to hold the data, again dropping the rows with NULL/NA values. We the created the target array from the is_hazardous column, and performed feature reduction to again remove the unecessary features (“name” and “orbiting_body”). The data and targets were then split into 80% training and 20% testing. With all of the data preprocessing complete, we then implemented the actual Logistic Regression model using the LogisticRegression class from scikit-learn. We used lbfgs as the solver (liblinear was considered due to smaller data sample but resulted in negligible differences in performance), and set maximum iterations to 500. We fit the model on the training data/labels, then used the trained model to predict the targets of the testing data. The predicted labels were then compared against the true test labels from the target array. To quantify/analyze the model’s performance, we again used basic scoring metrics, a confusion matrix, and 5-fold cross validation. Details and implications of the results are discussed in the Results Discussion portion for Logistic Regression.

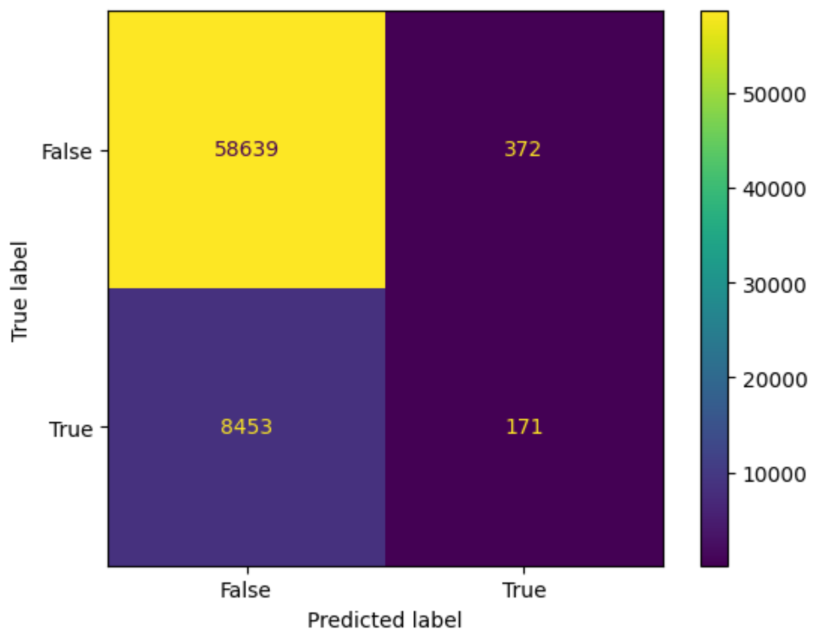
Gradient Boosting

Gradient Boosting was the final model we implemented. The implementation can be found in GradientBoost.ipynb in the Repository. The same initial setup was performed, with the dataset being read from the asteroids.csv file into a Pandas Dataframe and the NULL/NA rows being dropped. “name” and “orbiting_body” were dropped, and the is_hazardous column was used to populate the target array. A similar 80/20 split was used to create the training and testing data sets. We again chose to use scikit-learn for our model’s implementation, specifically using the GradientBoostingClassifier class from scikit-learn. When tuning the model parameters for optimal performance, we set n_estimators to 100 and learning_rate to 0.1 (commonly used settings), and max_depth equal to 3 since it is generally the default. We then fit the model on our training data, and generated predictions from the testing data. Again, evaluation of predicted labels was measured with basic scoring metrics, a confusion matrix, and 5-fold cross validation. See the Results and Discussion portion of Gradient Boosting for a more in depth discussion and explanation of these metrics and their implications.

Results and Discussion

We expect that these ML classification models to identify asteroids as potentially hazardous will be pretty accurate and will mostly rely on/weigh the closest distance to earth and velocity information. For quantitative metrics we’ll use to measure model performance, the first would be accuracy, as it is a simple metric to help us understand how well the model assigned predicted target values. A confusion matrix will also be useful, as seeing true positives, false positives, etc. is great for identifying classification model performance. Finally, cross validation is another metric we’ll use, as since we have such a large dataset we plan to sample, we can create multiple subsets of data and test for overfitting.

Naive Bayes - Results



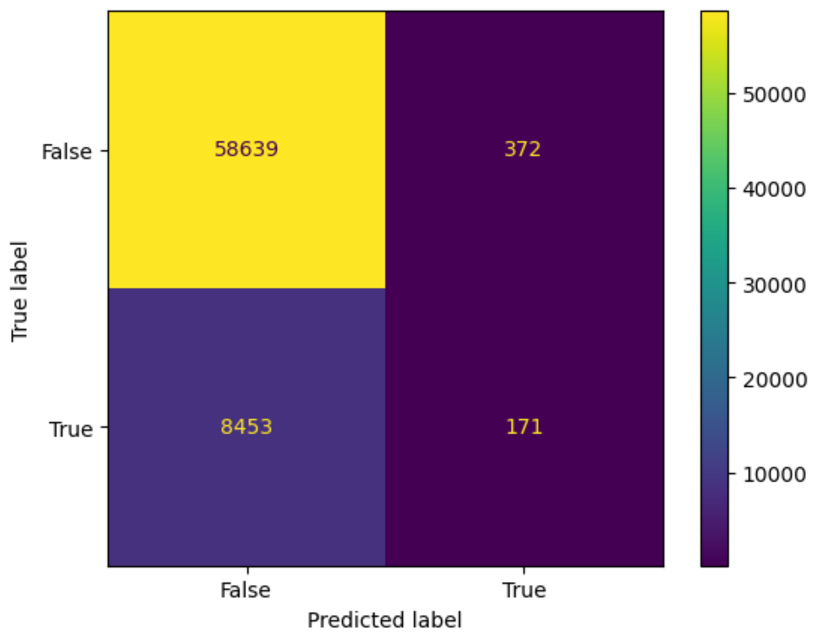
Basic scoring metrics:

- Accuracy: 0.867701633769498
- Precision: 0.30049261083743845
- Recall: 0.02829313543599258
- F1 Score: 0.05171682916490038

5-Fold Cross-Validation Accuracies: [0.86852961 0.86837981 0.8699175 0.87120383 0.8714404]

Looking at the 5-fold cross validation accuracies, we can see that the model is generally consistent and stable. However, looking at the rest of the gathered results, while we see a fairly high overall accuracy, we also have a relatively low precision, recall, and F1 score. The provided confusion matrix helps to explain this:while our Naive Bayes model is excellent at correctly identifying non-hazardous asteroids (true negative), it is generally ineffective at correctly identifying hazardous asteroids (true positives). Since about 87% of the data is non-hazardous (and thus only 13% being hazardous), we end up with a decently high accuracy score due to the large number of true negative identifications, but a low precision, recall, and F1 score due to the low amount of true positives and high amount of false negatives. The very low recall is particularly a problem, and not surprising given that Naive Bayes assumes features are completely independent, which may not be the case for an asteroid's luminosity, diameter, and velocity. In addition, Naive Bayes can often have a problem with overpredicting the most frequent class. Given the imbalanced nature of the dataset, while Naive Bayes may be useful for correctly identifying non-hazardous asteroids and simple to implement, the other methods may be more effective for correctly identifying hazardous asteroids (which is the more important part of the problem we want to solve).

Logistic Regression - Results



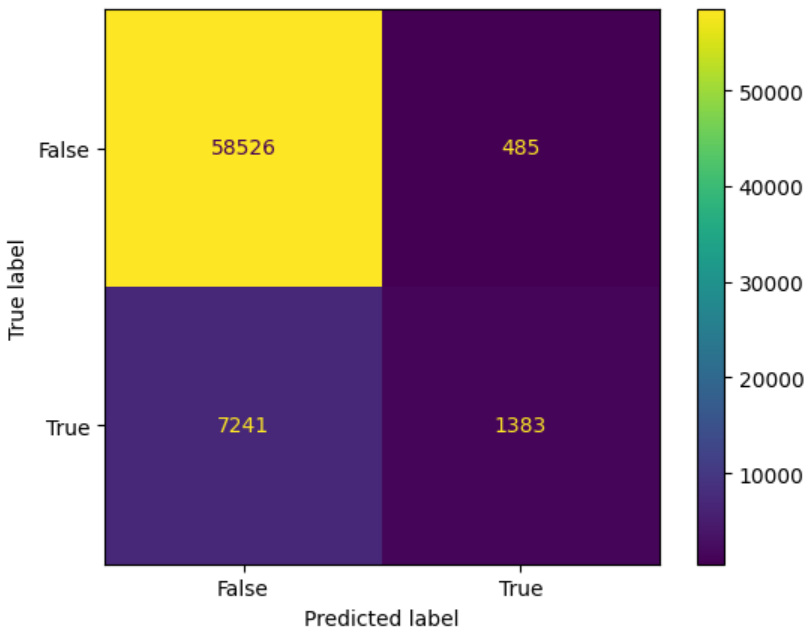
Basic scoring metrics:

- Accuracy: 0.869520218821616
- Precision: 0.3149171270718232
- Recall: 0.019828385899814447
- F1 Score: 0.037307734264208574

5-Fold Cross-Validation Accuracies: [0.87025948 0.87036106 0.87012449 0.86987314 0.86987314]

Looking at the 5-fold cross validation accuracies, again we can see that the model is generally consistent with and stable. The results, however, are nearly identical to those of Naive Bayes. This is probably because the relationship between our features is assumed to be independent for our Naive Bayes implementation, specifically Gaussian Naive Bayes. Logistic regression is also a linear separator, and Gaussian Naive Bayes could produce a result which looks like a linear boundary, which may be poor for this data set. Again, we ended up with a decently high accuracy of 87% of the asteroid due to around 87% of the data set being labeled as (non-hazardous). Again, a low precision, a very low recall, and F1 score due to the low amount true positives and high number of false negatives, but a high accuracy due to the large amount of true negative identifications. This is not a surprise; logistic regression is sub-optimal on highly imbalanced data sets (like ours), which especially is a problem for recall. It is as effective as Naive Bayes: to say, other methods might be more effective.

Gradient Boosting - Results



Basic scoring metrics:

- Accuracy: 0.8857692023360686
- Precision: 0.7403640256959315
- Recall: 0.16036641929499074
- F1 Score: 0.26362943194815097

5-Fold Cross-Validation Accuracies: [0.88341835 0.88440725 0.88464382 0.88601887 0.88850282]

Looking at the 5-fold cross validation accuracies, again we can see that the model is generally consistent with and stable. The results, however, are different that the previous two methods. This makes sense as Gaussian Boosting is better than the previous two methods simply due to the large imbalance in data. This time we ended up with a decently high accuracy of 88.5% of the asteroids, which is slightly more than the 87% of the data set being labeled as (non-hazardous): a lot of true negative identifications but also some true positives. The precision, notably, is roughly 70% which is a considerable improvement due to an increase in the number of true positives. The recall and F1 score are also considerably improved, going from very low to simply low. Roughly 3% to 16% from the first two methods for recall, and from 3%-5% to 26% for F1 score. Still very low, and I wouldn't trust all of humanity on this model. A higher precision doesn't mean much for asteroids: false positives can be double-checked by a human, false negatives for recall shouldn't be missed and could lead to a catastrophic situation. But there was an improvement over the previous two.

Final Conclusions

We found improved results as we progressed through the models. We expected to do better on the models, notably with gradient boosting, though even our best results were far from perfect. Our priority was to minimize false negatives in the confusion matrix and maximize recall which gradient boosting was able to achieve the best out of the three models we tested, though not to a sufficient effect for practical use, at least as is. In comparison to gradient boosting both logistic regression and Naive Bayes had very poor recall, F1, and notably worse precision, though precision is less of a priority for our ideal model. Given that gradient boosting produced significantly better results that other tested models going forward we could probably refine the parameters of gradient boosting to improve our F1 and most importantly recall to try and minimize false negatives. False positives on the other hand are less of a concern since, assuming the model sufficiently minimizes false positives, human intervention can be used to evaluate all positive results due to the imbalance in the true labels of the data set. This is fairly reasonable given the severity of the data being analyzed and the need for caution, however for this method to be effective we would need to significantly reduce the incidence of false negatives.

References

[1] Nemmaluri Purna Surekha, Preethi Nanjundan, and S. Bashir, “Prediction of Hazardous Asteroids Using Machine Learning,” International Conference on Emerging Systems and Intelligent Computing, pp. 169–174, Feb. 2024, doi: [https://doi.org/10.1109/esic60604.2024.10481589].

[2] Vedant Bahel, Pratik Bhongade, J. K. Sharma, S. Shukla, and Mahendra Gaikwad, “Supervised Classification for Analysis and Detection of Potentially Hazardous Asteroid,” Nov. 2021, doi: [https://doi.org/10.1109/iccica52458.2021.9697222].

[3] T. Sharma, S. Sharma, A. Sharma, A. Kumar, A. Malik, and A. Sharma, “Asteroid Hazard Prediction Using Machine Learning: A Comparative Analysis of Different Algorithms,” Nov. 2023, doi: [https://doi.org/10.1109/aece59614.2023.10428633].

Schedule and Contributions

Our semester task schedule can be found in this Gantt Chart

Proposal Contributions (entirety of project)

Name	Contributions
Evan S.	Wrote the writeup for Logistic Regression and Gradient Boosting ML methods. Helped with asset embeds and recorded some of the slides for the video.
Kerrick M.	Worked on coding for LogReg/GradientBoosting .ipynb’s, put together outline of final presentation slides and recorded the first few.
Mike A.	Dissected all the results, and gave reasoning for why the results were what they are. Recorded some of the slides for the video. Provided good next steps on what a model trained on this dataset should look like.
Owen M.	Wrote final conclusion edited and proofread slides, compiled recordings, produced and edited final presentation

About Repo

`/assets/` Stores media for use in the writeups

`/assets/nb_confusion_matrix.PNG` Confusion matrix created from our Naive Bayes implementations

`/assets/lr_confusion_matrix.PNG` Confusion matrix created from our Logistic Regression implementations

`/assets/gb_confusion_matrix.PNG` Confusion matrix created from our Gradient Boosting implementations

`/NaiveBayes.ipynb` Naive Bayes code for the project. Jupyter Notebooks for easy collaboration

`/LogRegression.ipynb` Logistic Regression code for the project. Jupyter Notebooks for easy collaboration

`/GradientBoosting.ipynb` Gradient Boosting code for the project. Jupyter Notebooks for easy collaboration

`/asteroids.zip` Raw asteroids data. CSV in a .zip file