Amrut Kulkarni akulkarni74
CS 4641 Final Project

Predicting Hazardousness Of Asteroids With Classification Algorithms

Introduction To Dataset

The dataset chosen was the NASA: Asteroids Classification— available at this Kaggle.com link https://www.kaggle.com/shrutimehta/nasa-asteroids-classification, and it consists of 4688 asteroids classified by NASA with 40 different attributes per asteroid that can be used to determine whether each one may be hazardous/may impact Earth at some point in the future. This topic is important for obvious reasons; if it is possible to predict which asteroids could be hazardous to Earth, and by extension the continuing existence of humanity, then it is important to build models that could do that so that these asteroids can be identified and humanity can have measure in place in advance of a potential impact event. This topic is interesting to me because I grew up very interested in astronomy and astrophysics, so virtually any topic related to space, cosmology, etc. will be incredibly interesting to me. The metrics that will be used to evaluate each algorithm's performance with this dataset will be the calculated testing scores, after hyperparameters are found through Grid Search using 5-fold cross validation, and the time it takes to tune each model.

Introduction To Algorithms

Random Forest With Bagging

A Random Forest Classifier (RFC) that incorporates Bagging will train *n* number of decision tree classifiers on random subsets of the training data, splitting each learner on random features in the training data, and the learners will be combined so that the resulting metal-earner will have stronger performance than any of the individual learners.

The implementation used for this project was the RandomForestClassifier from Sci-kit Learn— this model is implemented with Bagging as long as the parameter bootstrap Is True, which it is whenever this model is used for this project

Support Vector Machine

A Support Vector Machine (SVM) is a learner that can project a dataset in such a way that it can find an optimal boundary between the different classes of that dataset using an appropriate kernel function.

The implementation used for this project was the SVC from Sci-kit Learn

Neural Net

A Neural Net (NN) is a learner that takes in data in its input layer, processes features of the data in multiple layers of neurons/perceptrons that find patterns in the data, and each instance of data is classified in the output layer.

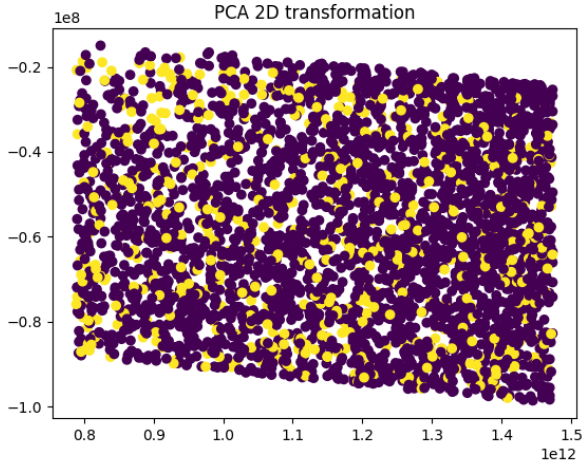
The implementation used for this project was the MLPClassifier from Sci-kit Learn

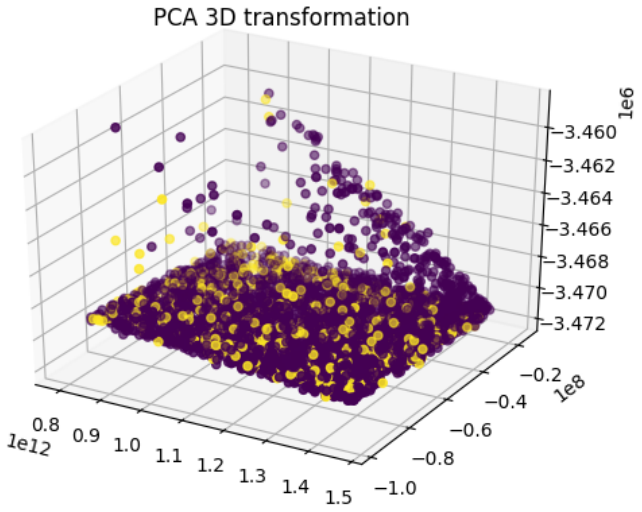
Tuning Hyperparameters

Data Distribution

The three requirements to determine if a dataset has a non-trivial distribution is that it needs to have at least 1000 instances, which each need to have at least 5 attributes, that the models can learn from, and it can not be linearly separable. The NASA asteroid dataset fulfills the first two requirements since it has 4688 instances of asteroids, and each asteroid has 40 attributes, of which 22 were used to train and test the different algorithms. To show that the dataset is not linearly separable, the PCA

transformation was applied to it and it was plotted in both 2D and 3D. In order for a dataset to be linearly separable, a linear hyperplane must be able to divide the different classes of data into mostly exclusive groups, but both the 2D and 3D plots show that both classes of data heavily intermingle to the point where a linear hyperplane won't be able to divide them effectively.





Random Forest Tuning

```
==== Tuning RFC with Bagging ====
Test Score:  0.9936034115138592  with parameters:  {'max_depth': 10000, 'n_estimators': 10}
Total time to run Grid Search:  381.30869483947754  seconds

Generating Table
    max_depth  n_estimators  Mean Train Score  Mean Test Score  Rank
0         100            10          0.999667         0.994932    11
1         100           100          1.000000         0.995999     3
2         100          1000          1.000000         0.995999     4
3         100         10000          1.000000         0.995732     6
4        1000            10          0.999667         0.994399    12
5        1000           100          1.000000         0.996267     2
6        1000          1000          1.000000         0.995732     6
7        1000         10000          1.000000         0.995732     6
8       10000            10          0.999933         0.996799     1
9       10000           100          1.000000         0.995999     4
10      10000          1000          1.000000         0.995466    10
11      10000         10000          1.000000         0.995732     6
```

In order to find the optimal hyperparameters for the RFC, the model was processed using GridSearchCV, with the default 5-fold cross validation, to test different maximum depths and number of estimators/decision tree learners. Maximum depth determines how far-spanning the final meta-learner will be—if there is no max depth, the learner is at risk of over-fitting the data, and n_estimators determines how many individual tree learners were combined to find the final meta learner.

The different hyperparameters tested for the RFC were 10, 100, 1000, 10000 for the number of estimators and 100, 1000, 10000 for the maximum depth of the resulting Random Forest

In this case, the optimal max depth was 10000 and the optimal n_estimators was 10 estimators. The GridSearch took 338.076 seconds to run to completion.

SVM Tuning

```
==== Tuning SVM ====
Test Score:  0.8454157782515992  with parameters:  {'C': 0.001, 'kernel': 'poly'}
Total time to run Grid Search:  275.80213713645935  seconds

Generating Table
         C   kernel  Mean Train Score  Mean Test Score  Rank
0    0.001     poly          0.837290         0.837290     1
1    0.001      rbf          0.837290         0.837290     1
2    0.001  sigmoid          0.837290         0.837290     1
3    0.100     poly          0.837290         0.837290     1
4    0.100      rbf          0.837290         0.837290     1
5    0.100  sigmoid          0.837290         0.837290     1
6    1.000     poly          0.837290         0.837290     1
7    1.000      rbf          0.837290         0.837290     1
8    1.000  sigmoid          0.772802         0.775690    15
9   10.000     poly          0.837290         0.837290     1
10  10.000      rbf          0.837290         0.837290     1
11  10.000  sigmoid          0.726059         0.730334    16
12 100.000     poly          0.837290         0.837290     1
13 100.000      rbf          0.837290         0.837290     1
14 100.000  sigmoid          0.721325         0.727665    17
15 1000.000     poly          0.837290         0.837290     1
16 1000.000      rbf          0.837290         0.837290     1
17 1000.000  sigmoid          0.721191         0.727398    18
```

In order to find the optimal hyperparameters for the SVM Classifier, the model was processed using GridSearchCV, with the default 5-fold cross validation, to test different values of C and different kernel functions.

The different hyperparameters tested for SVM were 0.001, 0.1, 1, 10, 100, 1000 for values of C and 'poly', 'rbf', 'sigmoid' for different kernel functions

In this case, the optimal value of C waas 0.001 and the optimal kernel was polynomial

The GridSearch took 275.8 seconds to run to completion.

Neural Net Tuning

```
==== Tuning NN ====
Test Score:  0.8454157782515992  with parameters:  {'hidden_layer_sizes': (3, 4), 'learning_rate': 'constant'}
Total time to run Grid Search:  131.8554549217224  seconds

Generating Table
   hidden_layer_sizes learning_rate  Mean Train Score  Mean Test Score  Rank
0              (3, 4)      constant          0.837290         0.837290     1
1              (3, 4)    invscaling          0.702357         0.702443    18
2              (3, 4)      adaptive          0.837290         0.837290     1
3              (4, 3)      constant          0.702378         0.702357    21
4              (4, 3)    invscaling          0.837290         0.837290     1
5              (4, 3)      adaptive          0.837290         0.837290     1
6              (5, 5)      constant          0.837290         0.837290     1
7              (5, 5)    invscaling          0.702378         0.702357    21
8              (5, 5)      adaptive          0.837290         0.837290     1
9           (5, 5, 5)      constant          0.837290         0.837290     1
10          (5, 5, 5)    invscaling          0.837290         0.837290     1
11          (5, 5, 5)      adaptive          0.702378         0.702357    21
12             (6, 6)      constant          0.702357         0.702443    18
13             (6, 6)    invscaling          0.702378         0.702357    21
14             (6, 6)      adaptive          0.837290         0.837290     1
15          (6, 6, 6)      constant          0.837290         0.837290     1
16          (6, 6, 6)    invscaling          0.837290         0.837290     1
17          (6, 6, 6)      adaptive          0.837290         0.837290     1
18             (7, 7)      constant          0.702378         0.702357    21
19             (7, 7)    invscaling          0.837290         0.837290     1
20             (7, 7)      adaptive          0.432533         0.432577    30
21          (7, 7, 7)      constant          0.837290         0.837290     1
22          (7, 7, 7)    invscaling          0.567467         0.567423    29
23          (7, 7, 7)      adaptive          0.837290         0.837290     1
24             (8, 8)      constant          0.837290         0.837290     1
25             (8, 8)    invscaling          0.702378         0.702357    21
26             (8, 8)      adaptive          0.837290         0.837290     1
27          (8, 8, 8)      constant          0.702378         0.702357    21
28          (8, 8, 8)    invscaling          0.702357         0.702443    18
29          (8, 8, 8)      adaptive          0.702378         0.702357    21
```

In order to find the optimal hyperparameters for the Neural Net Classifier, the model was processed using GridSearchCV, with the default 5-fold cross validation, to test different amounts of hidden layers, number of neurons per layer, and different learning rates.

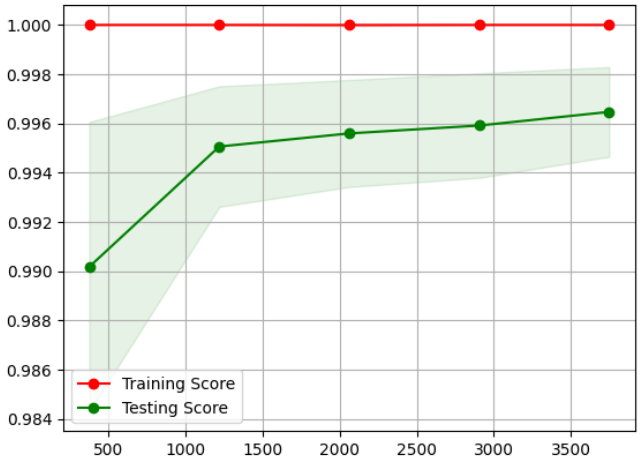
The different hyperparameters tested for the Neural net were 2-3 hidden layers with between 3 and 8 neurons per layer—(3, 4), (4, 3), (5, 5), (5, 5, 5), (6, 6), (6, 6, 6), (7, 7), (7, 7, 7), (8, 8), (8, 8, 8), where each number is a tuple entry is a layer consisting of that many neurons, and "constant", "inverse scaling", and "adaptive" for possible learning rates

In this case, the optimal hidden layers and neurons per layer were 2 hidden layers with 3 neurons in the first layer and 4 neurons in the second layer—(3,4), and the optimal learning rate was constant.

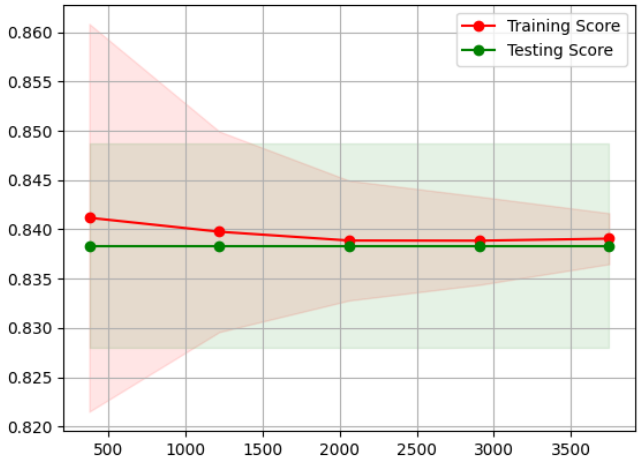
The GridSearch took 132 seconds to run to completion.

Comparing Optimized Model Performances

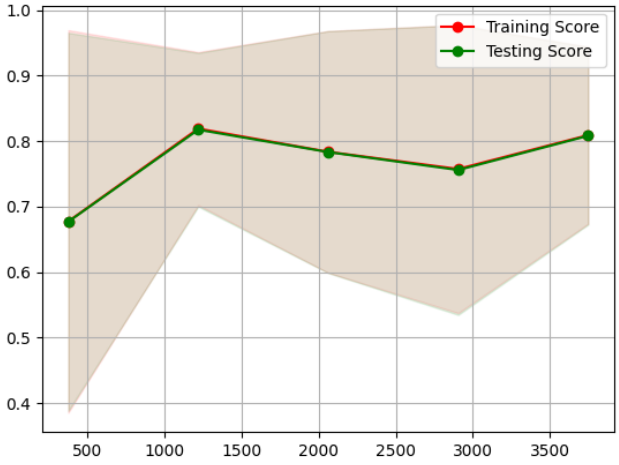
Random Forest Performance



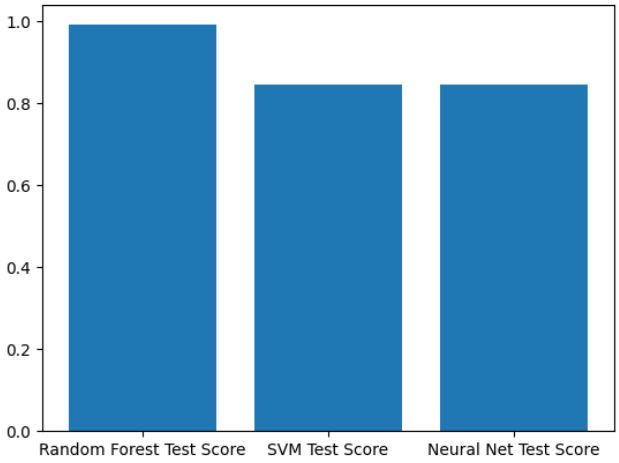
Support Vector Machine Performance



Neural Net Performance



Testing Score Comparison



Out of the three different models, RFC, Random Forest with Bagging, performs the strongest of the three and Neural Nets perform the weakest of the three. However, all three models had scores above 80% on the test data, so each would make a good classifier for this dataset. However, given the importance of the subject, RFC would be recommended.

Conclusion

In terms of performance, RFC would be the most preferable dataset, since it was able to predict whether an asteroid was potentially hazardous over 99% of the time. This could be because, with the depth that this tree was able to grow to, as well as being the average of the results of multiple learners, it is very likely that there are paths from root node to leaf node that are partitioned along very specific differences between instances in the dataset, which would be necessary since it appears that there isn't that much difference between a hazardous and non-hazardous asteroid.

In terms of time to find optimal hyperparameters, the Neural Net took the least amount of time to tune, but its performance is considerable lower than that of the RFC. This could also be because there differences in attributes between a non-hazardous and hazardous asteroid are too small for sklearn's MLPClassifier to pick up on in the same way the RFC can, so that even though the NN can get up to 84.5% accuracy, it can't predict at a higher accuracy with this NN implementation.

The SVM was in between both the RFC and NN in terms of performance and time needed to tune its hyperparameters. It is possible that since both classes of asteroid are very similar, and as the 2D and 3D plots show, very closely grouped, there isn't a kernel function that can project both classes in a way that they can be separated on an optimal boundary in more tha around 84% of cases.