##
# Author: Michael Wood
# Purpose: Technical Report for Assignment 7: NASA Asteroid Classification Problem, create an MLPClassifier model that can classify an asteroid as Hazardous or Non-Hazardous (binary classification) as accurately as possible.
# Code of assignment in HW7_Python.ipynb under AI repository in HW7 folder in Github
# Class: AI, CSC 362, Spring 2024
##

Preparing the Data:

In order to create an MLPClassifier model, I first had to prepare and load the dataset. I loaded all the necessary libraries first, and loaded the data set with the nasa_asteroid.csv file. I added the AI_ML_Datasets folder provided from the Assignment into my own Google Drive. I then mounted the Google Drive and used the path that led to the needed file.

The next step was to split the dataset into train and test sets. For this part, I first defined the 'X' and 'y' variables for the data sets. For 'X', I examined the dataset and dropped any non-relevant features such as 'Name', 'Orbit ID', 'Close Approach Date', etc. Only the 'Hazardous' values were stored in 'y' since that was what the asteroids were being classified as. Once 'X' and 'y' were defined, I split the data into train and test sets using the 70%-30% rule. I also randomly shuffled the data and stratified it using 'y'.

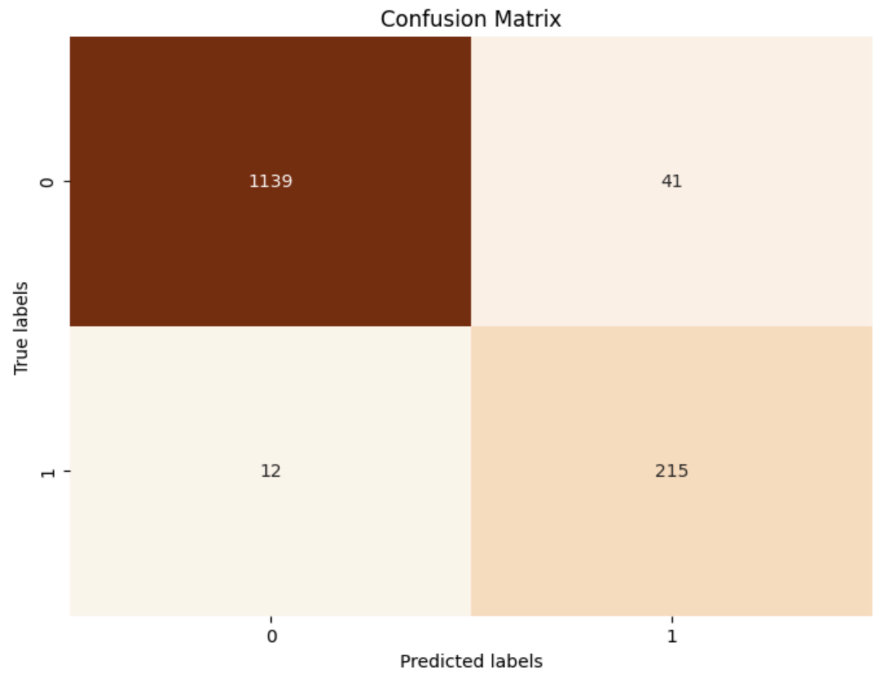
The MLPClassifier:

For the actual classification, I initially used two hidden layers with 50 neurons for each of them. Next, I initialized two lists to store accuracies for both the training and test sets, respectively. Then, I scaled the numeric features of the data using StandardScaler and the code snippet provided on Canvas. Finally, I trained the classifier over 1000 iterations using the partial_fit() command. This snippet of code is similar to the training method used in the Iris_MLP_Classifer.ipynb file provided on Canvas. I also used some of the code in the file for displaying the overall accuracies on the dataset, the confusion matrix, the accuracy plots, and the loss curve.

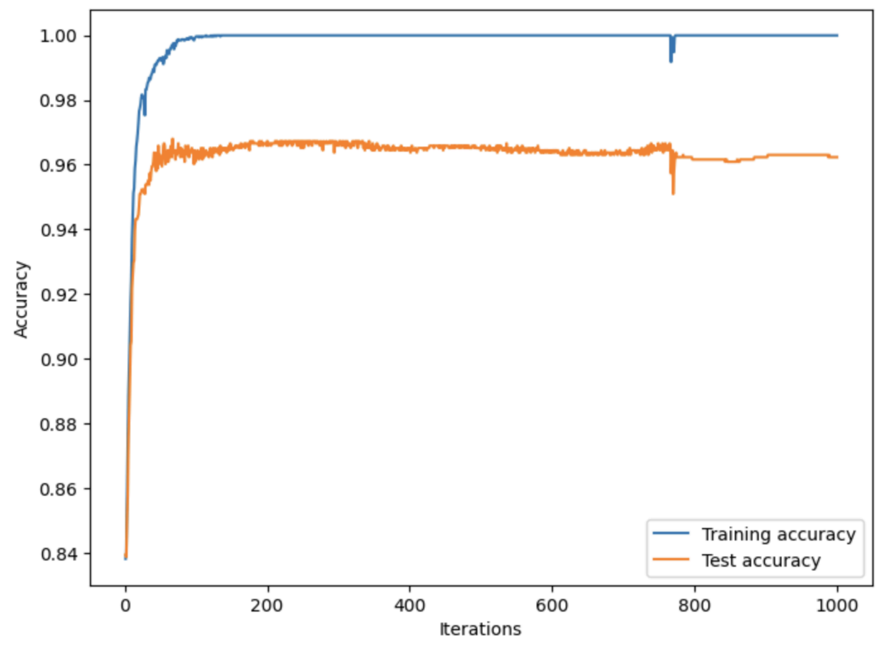
Results:

For this particular classification training, the accuracy on the train set was 1.0 and accuracy on the test set was 0.9623. Below are screenshots of the results for the corresponding confusion matrix on the test set, the accuracy plots, and the lost curve.

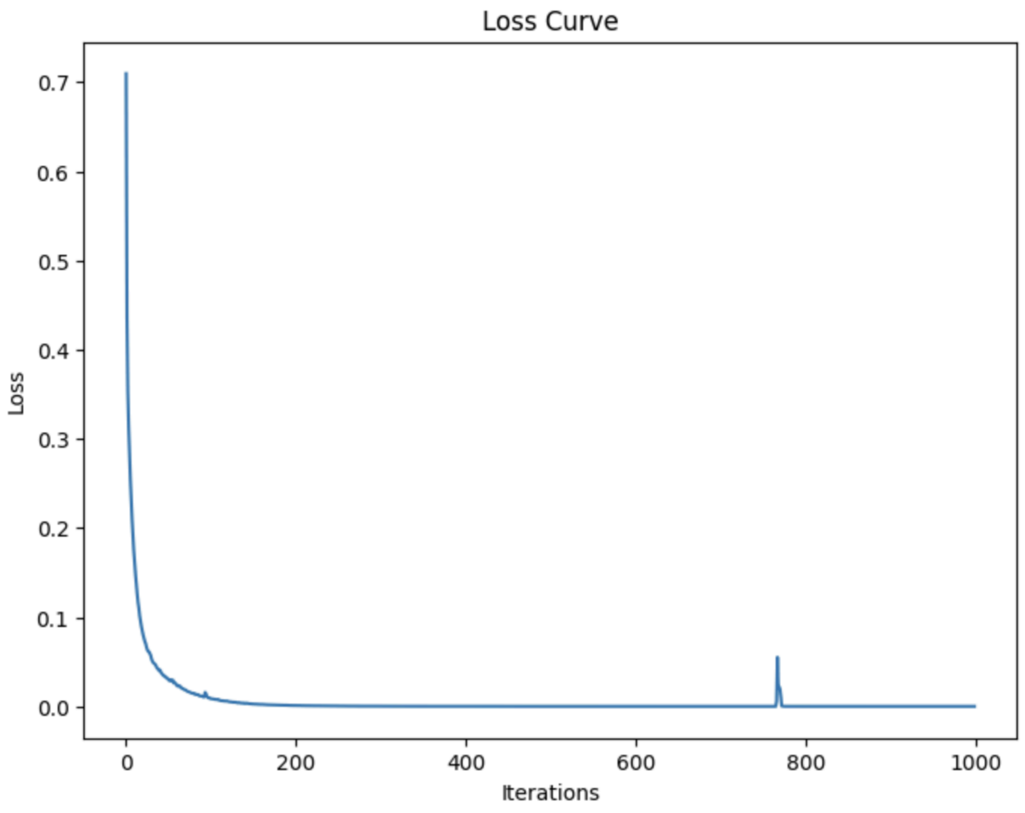
Confusion Matrix:



Train/Test Accuracy Plot:



Loss Curve:



Analysis:

The classifier was mostly accurate on both models with only some misclassifications for both hazardous and non-hazardous asteroids on the test set. The confusion matrix shows that it correctly classified 215 hazardous asteroids and 1,139 non-hazardous asteroids while misclassifying 12 hazardous asteroids and 41 non-hazardous asteroids. On both the accuracy plots and the loss curve, there was a bigger difference between the predicted values and the actual values than usual around the 800th iteration. The model was still able to maintain its normal accuracy beyond that point.

Different Choice of Hyperparameters:

While a test accuracy of around 96% is good, it is important to try to find as perfect accuracy as possible for the model to be 100% efficient. Different hyperparameters such as the number of hidden layers, the number of neurons in each hidden layer, and the activation function can be changed in order to try to find a more accurate model. I found the accuracies, confusion matrices, accuracy plots, and loss curves for all of the models. Below is a table of different hyperparameters used to try and achieve this goal. It displays the changes involved, the train and test accuracies, and the confusion matrix for each classification.

Table of Results:

Hyperparameters	**Train Accuracy**	**Test Accuracy**	**Confusion Matrix**
(Initial Classification) 2 hidden layers; 50 neurons for each	1.0	0.9623	[[1139 41] [12 215]]
(Changing number of neurons in each hidden layer) 2 hidden layers; 40 and 30 neurons, respectively	1.0	0.9716	[[1153 27] [13 214]]
(Changing the number of hidden layers) 3 hidden layers; 50 neurons each	1.0	0.9694	[[1149 31] [12 215]]
(Changing both the number of hidden layers and the number of neurons in each hidden layer) 3 hidden layers; 75, 60, and 45 neurons, respectively	1.0	0.9716	[[1152 28] [12 215]]
(Default parameter) 1 hidden layer; With 100 neurons	1.0	0.9694	[[1148 32] [11 216]]

(Changing the activation function)
2 hidden layers;
50 neurons each;
Sigmoid activation function
	1.0	0.9758	[[1159 21]
			[13 214]]

(Changing number of neurons in each layer and the activation function)
2 hidden layers;
30 and 20 neurons, respectively;
Sigmoid activation function
	1.0	0.9815	[[1165 15]
			[11 216]]

(Having a small number of neurons in hidden layers) 2 hidden layers; 15 neurons each	1.0	0.9709	[[1154 26] [15 212]]
(Having more hidden layers with more neurons) 5 hidden layers; 100, 80, 60, 40, and 20 neurons in each, respectively	1.0	0.9687	[[1147 33] [11 216]]

Analysis:

It appears that none of these models achieved 100% accuracy on the test set, but they were all more efficient than the initial classification. The model that provided the best accuracy of 98% was the one with 2 hidden layers with 20 and 30 neurons in each hidden layer, respectively. The activation function was also changed to ‘logistic’ or sigmoid. This was to be expected since the sigmoid activation function works well with binary classification problems. It was even one of the models that was able to correctly classify more hazardous asteroids than the initial model, which again signified it was a more efficient classification. None of these models were able to achieve 100% accuracy, so more adjustments and training may be needed to have full efficient classification.