CSE 587 DATA INTENSIVE COMPUTING

PHASE 2

DETECTION OF IMPACT ZONE ON EARTH BY SPACE OBJECTS

Team members

Jahnavi Rudraraju

UBID: 50464467

jahnavir@buffalo.edu

Sujith Varma Vatsavai

UBID: 50442317

sujithva@buffalo.edu

Dataset Link: https://www.kaggle.com/datasets/sameepvani/nasa-nearest-earth-objects

Recap from Phase 1:

During phase 1 we have preprocessed the code using data cleaning methods such as Outliers, splitting columns, Renaming (or) clear formatting, Duplicate Entries, Missing values, Inconsistent values, Data transformation. Exploratory Data Analysis(EDA) is used for visualization of the dataset.

Introduction:

We used a data cleaning method during model training on the hazardous column the method is to assign 0 and 1 to the values of the data-frame column, after that we split the total data into training and testing data. We use various algorithms for classification of the data.

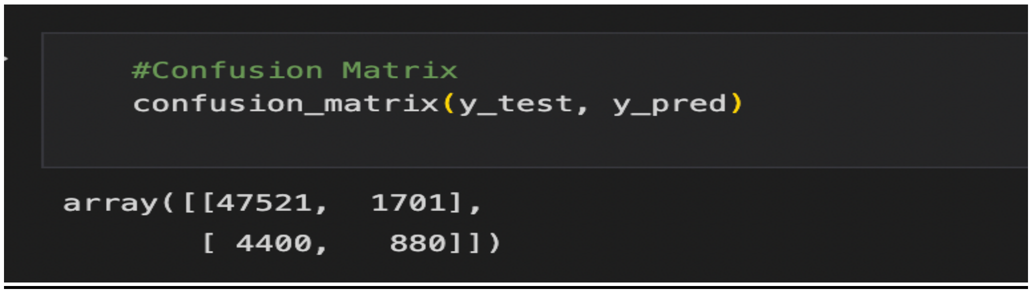
Algorithms & Analysis:

K-Nearest Neighbour:

The supervised learning technique K-nearest neighbors (KNN) is used for both regression and classification. By calculating the distance between the test data and all the training points, KNN tries to predict the correct class for the test data. Then select the K number of points which is close to the test data. The KNN algorithm calculates the probability of the test data belonging to the classes of 'K' training data and class holds the highest probability will be selected.

Output:

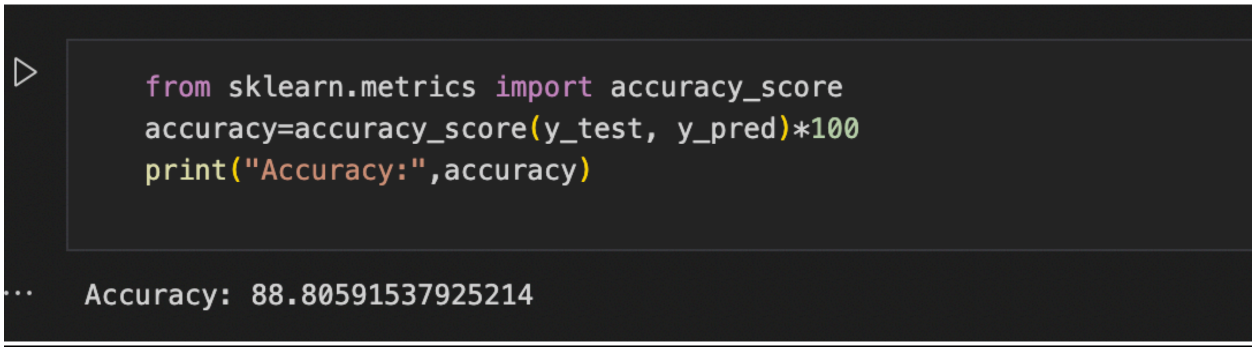
Confusion Matrix :



```
#Confusion Matrix
confusion_matrix(y_test, y_pred)
```

```
array([[47521,  1701],
       [ 4400,   880]])
```

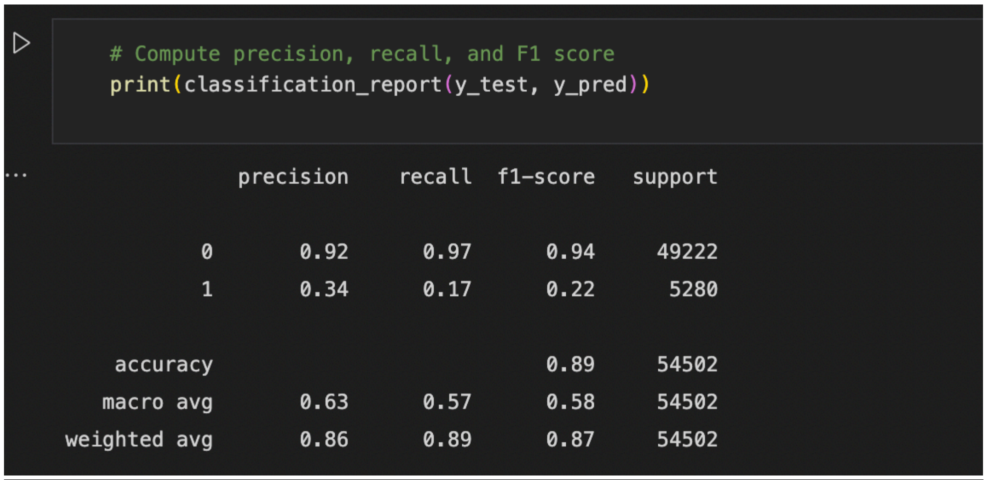
Accuracy :



```
from sklearn.metrics import accuracy_score
accuracy=accuracy_score(y_test, y_pred)*100
print("Accuracy:",accuracy)
```

```
Accuracy: 88.80591537925214
```

Report :



```
# Compute precision, recall, and F1 score
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.94     49222
           1       0.34      0.17      0.22      5280

    accuracy                           0.89     54502
   macro avg       0.63      0.57      0.58     54502
weighted avg       0.86      0.89      0.87     54502
```

Analysis and Reason for using this algorithm:

This algorithm can be effective for classification tasks when there is a relatively small number of features and a large amount of training data, as it does not require any explicit training process. Additionally, KNN is a non-parametric method, meaning that it does not assume any specific distribution for the data. As our NASA dataset have relatively small number of features and the relationships between them are relatively simple. So, it seemed to be working efficiently.

Advantages:

1) KNN modeling is very time-efficient in terms of improvisation for a random modeling on the available data because it does not require a training period as the data itself is a model that will serve as the reference for future prediction.
2) KNN is very easy to implement. The only thing that needs to be calculated for KNN is the distance between various points using data from various features, and this distance can simply be calculated using distance formulas like Euclidian or Manhattan distances.

Disadvantages:

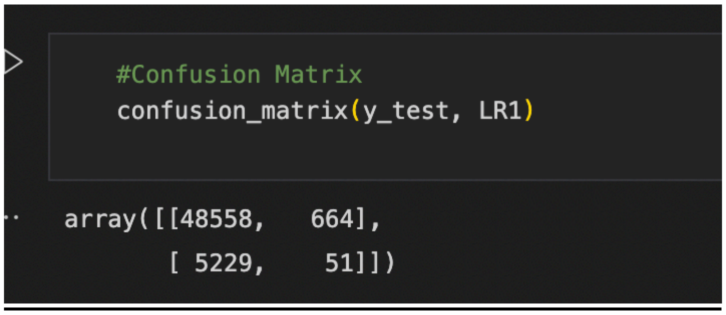
1. Does not work well with large datasets since it would be exceedingly expensive to calculate the distances between each data item.
2. Does not work well with large dimensionality as this will make computing distance for each dimension more difficult.
3. Data should be properly scaled (normalized and standardized) across all dimensions.

Logistic Regression:

Logistic regression is a simple, yet effective algorithm for binary classification tasks. It models the probability of the target variable given the features and can be trained efficiently on large datasets. Logistic regression estimates the probability of an event occurring, based on a given dataset of independent variables.

Output:

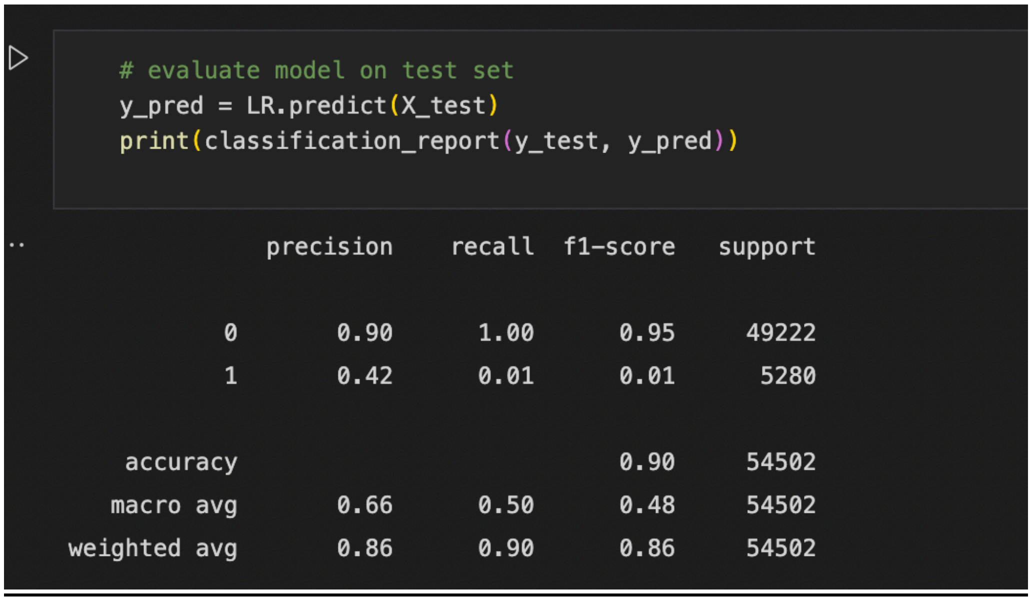
Confusion Matrix :



```
#Confusion Matrix
confusion_matrix(y_test, LR1)
```

```
array([[48558,   664],
       [ 5229,    51]])
```

Report:



```
# evaluate model on test set
y_pred = LR.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95     49222
           1       0.42      0.01      0.01      5280

    accuracy                           0.90     54502
   macro avg       0.66      0.50      0.48     54502
weighted avg       0.86      0.90      0.86     54502
```

Analysis and Reason for using this algorithm:

Logistic regression is a linear model that can handle both binary and multi-class classification problems. Since our NASA dataset has a binary target variable(if it is hazardous or not), this algorithm is a good choice.

Advantages:

1) Outputs have a nice probabilistic interpretation, and the algorithm can be regularized to avoid overfitting.
2) Logistic models can be updated easily with new data using stochastic gradient descent.

Disadvantages:

1) Logistic regression tends to underperform when there are multiple or non-linear decision boundaries.

Naïve Bayes: Naive Bayes is a classification algorithm that is based on Bayes' theorem and the assumption of conditional independence between features. Naive Bayes assumes that the presence or absence of a particular feature in a class is independent of the presence or absence of any other feature in that class. This assumption simplifies the calculation of the conditional probabilities needed for classification. The algorithm works by calculating the probability of each class given the input features, and then selecting the class with the highest probability.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

A is the hypothesis that a tuple B belongs to class C.

A is the Class Posterior Probability.

P(A) is the Class Prior Probability.

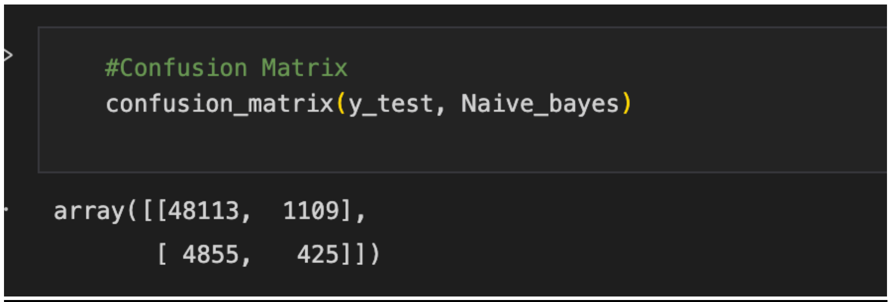
P(B/A) is the Descriptor Posterior Probability.

P(B) is the Descriptor Prior Probability

Note: This algorithm assumes that every attribute is independent of each other, and this assumption is known as class conditional Independence.

Output:

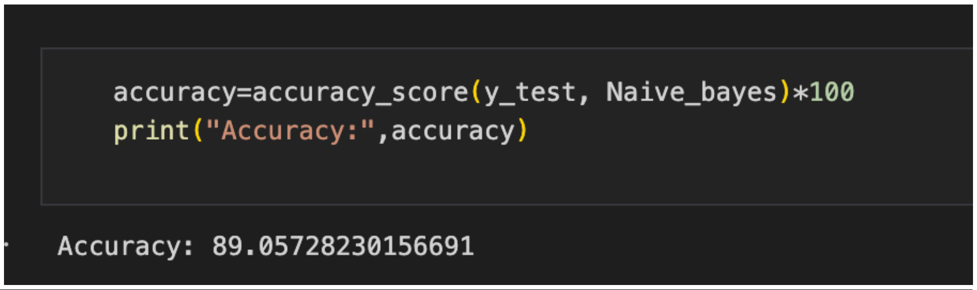
Confusion Matrix:



```
#Confusion Matrix
confusion_matrix(y_test, Naive_bayes)
```

```
array([[48113,  1109],
       [ 4855,   425]])
```

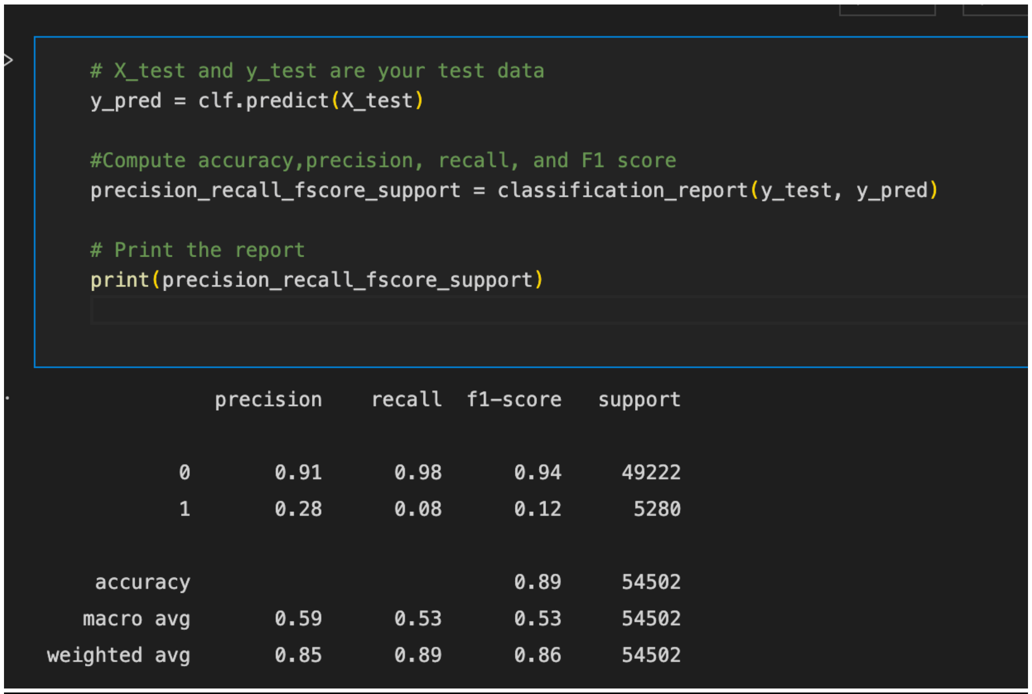
Accuracy :



```
accuracy=accuracy_score(y_test, Naive_bayes)*100
print("Accuracy:",accuracy)
```

```
Accuracy: 89.05728230156691
```

Report:



```
# X_test and y_test are your test data
y_pred = clf.predict(X_test)

#Compute accuracy,precision, recall, and F1 score
precision_recall_fscore_support = classification_report(y_test, y_pred)

# Print the report
print(precision_recall_fscore_support)
```

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94     49222
           1       0.28      0.08      0.12      5280

    accuracy                           0.89     54502
   macro avg       0.59      0.53      0.53     54502
weighted avg       0.85      0.89      0.86     54502
```

Analysis and Reason for using this algorithm:

Naive Bayes can handle both numerical and categorical data, and it is especially useful for high-dimensional datasets. It can also handle missing values and noisy data, and it is relatively robust to irrelevant features. As we have used high dimensional dataset which also contains numerical and categorial data it made us to use this classification algorithm.

Advantages:

1) This algorithm handles both continuous and discrete data.
2) It is insensitive to irrelevant features.
3) For implementation, it is easier.
4) When attributes are independent to each other Naive Bayes gives apt results.

Disadvantages:

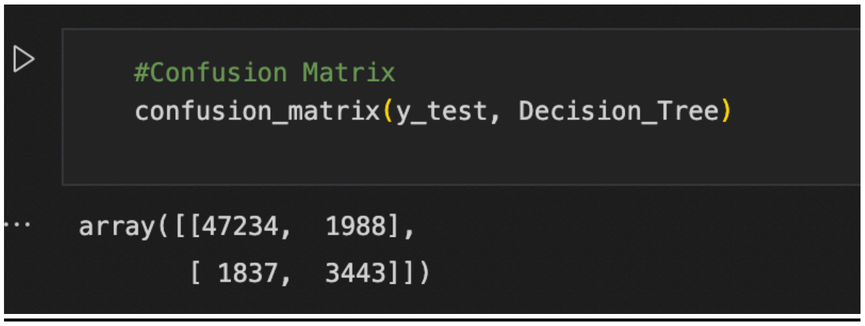
1) In Real-world, the usage of this algorithm is limited because Naive Bayes assumes that all predictors are independent.
2) If a categorical variable has a category in the test dataset, which was not observed in the training dataset, then the model will assign a 0 (zero) probability and will be unable to make a prediction.

Decision Tree:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. We perform split based on a particular attribute at each node. The leaf don't split and they are decision nodes. An optimal decision is made at each node to select a condition and a feature to perform the split.

Output :

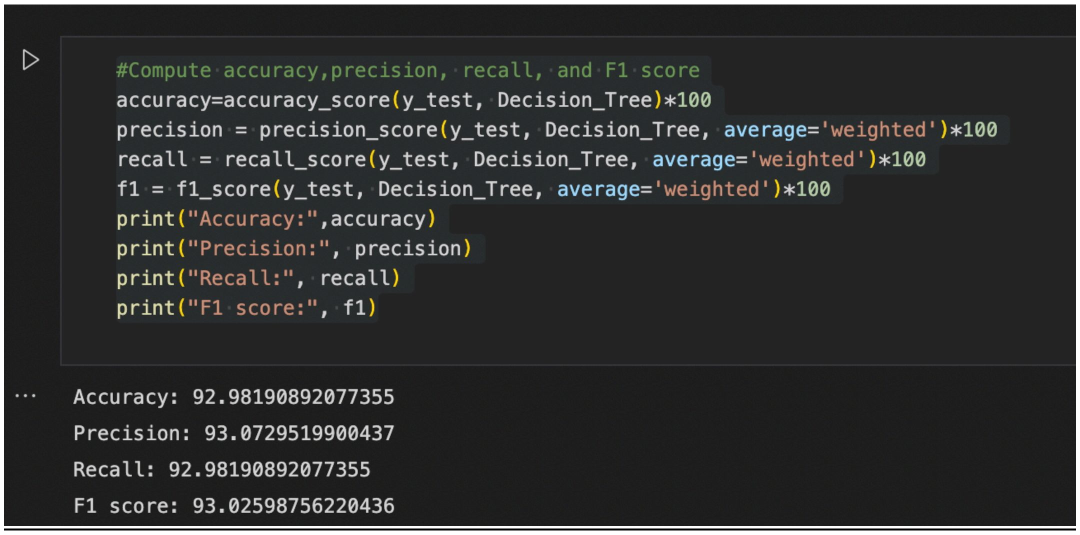
Confusion Matrix :



```
#Confusion Matrix
confusion_matrix(y_test, Decision_Tree)
```

```
array([[47234,  1988],
       [ 1837,  3443]])
```

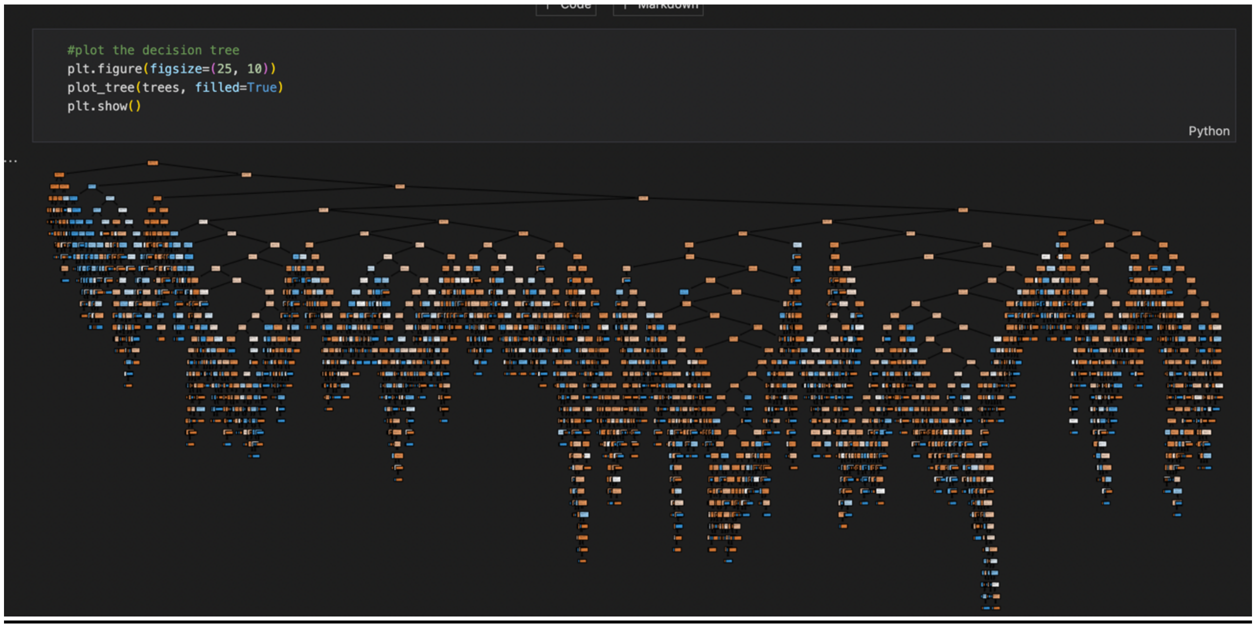
Report :



```
#Compute accuracy,precision, recall, and F1 score
accuracy=accuracy_score(y_test, Decision_Tree)*100
precision = precision_score(y_test, Decision_Tree, average='weighted')*100
recall = recall_score(y_test, Decision_Tree, average='weighted')*100
f1 = f1_score(y_test, Decision_Tree, average='weighted')*100
print("Accuracy:",accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
```

```
Accuracy: 92.98190892077355
Precision: 93.0729519900437
Recall: 92.98190892077355
F1 score: 93.02598756220436
```

Plot :



```
#plot the decision tree
plt.figure(figsize=(25, 10))
plot_tree(trees, filled=True)
plt.show()
```

Analysis and Reason for using this algorithm:

Decision trees are a popular choice for classification tasks because they are easy to understand and interpret. They can also be effective when there are complex, nonlinear relationships between features and the target variable. NASA dataset has a relatively small number of features which made us to choose this algorithm.

Advantages:

1) During preprocessing, data preparation requires less effort.
2) The process of creating a decision tree is not significantly impacted by missing values in the data.
3) There is less requirement of data cleaning compared to other algorithms.

Disadvantages:

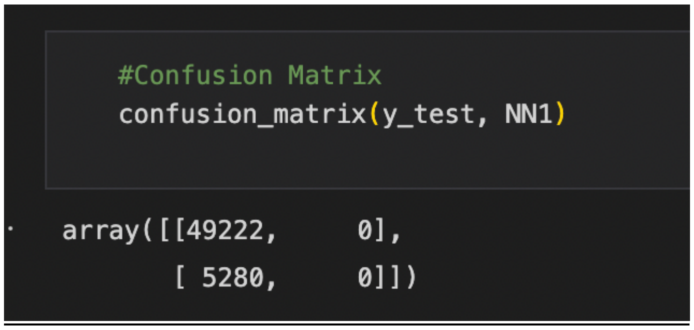
1) A slight change in the data can result in a big change in the decision tree's structure, which can lead to instability.
2) For application of regression and predicting continuous values this algorithm is inadequate.
3) For application of regression and predicting continuous values this algorithm is inadequate.
4) It takes more time to train the model.
5) For more class labels, the computational complexity of the decision tree may increase.

Neural Networks:

Neural networks are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. Neural networks can be effective for a wide range of classification tasks, including those with complex, nonlinear relationships between features and the target variable. They can be particularly effective when there is a large amount of training data available.

Output :

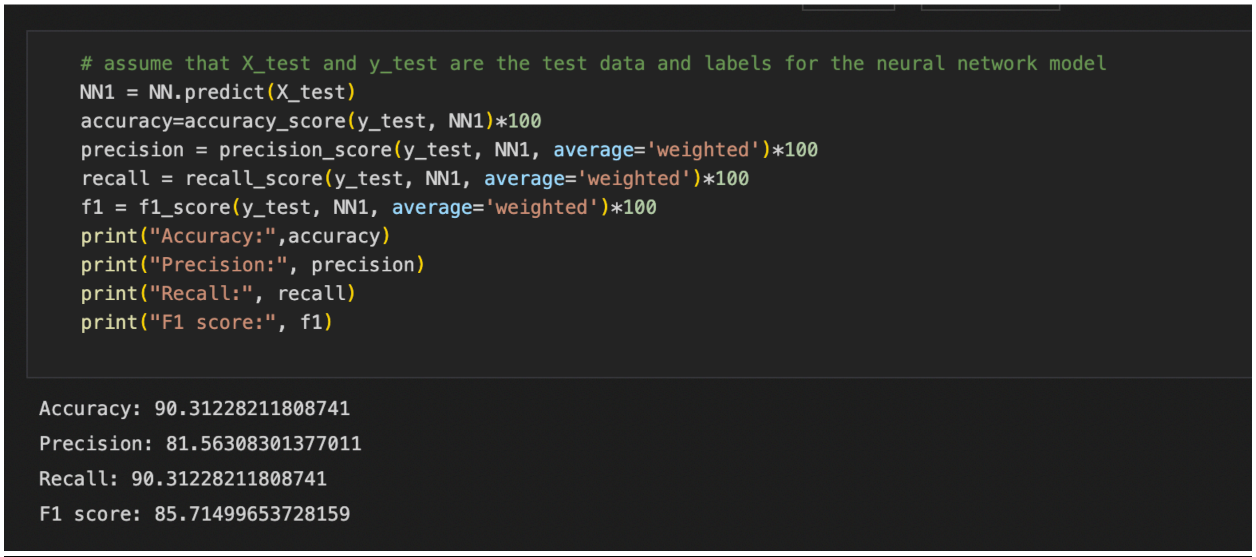
Confusion Matrix :



```
#Confusion Matrix
confusion_matrix(y_test, NN1)
```

```
array([[49222,     0],
       [ 5280,     0]])
```

Report:



```
# assume that X_test and y_test are the test data and labels for the neural network model
NN1 = NN.predict(X_test)
accuracy=accuracy_score(y_test, NN1)*100
precision = precision_score(y_test, NN1, average='weighted')*100
recall = recall_score(y_test, NN1, average='weighted')*100
f1 = f1_score(y_test, NN1, average='weighted')*100
print("Accuracy:",accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 score:", f1)
```

```
Accuracy: 90.31228211808741
Precision: 81.56308301377011
Recall: 90.31228211808741
F1 score: 85.71499653728159
```

Analysis and Reason for using this algorithm:

As our NASA dataset has many features and also, they are complex, and there are nonlinear relationships between them, neural networks is the perfect choice.

Advantages :

1) Can handle redundant and irrelevant attributes because weights are automatically learnt for all attributes.
2) Testing the data is extremely fast.

Disadvantages :

1) Sensitive to noise in training data
2) Difficult to handle missing attributes.
3) Multilayer ANN are universal approximators but could suffer from overfitting if the network is too large.

References:

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB

https://scikit-learn.org/stable/modules/neural_networks_supervised

https://scikit-learn.org/stable/modules/tree

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression

https://scikit-
learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier