**Intel Training Program**

**Fake News Detection using Python and Machine Learning**

**Team : Neural Nexus**

**Problem Statement:**

Sarah, an active social media user, is committed to sharing accurate and reliable information. To achieve this, she uses a fake news detector tool. One day, she comes across an alarming article shared by a friend. Sarah copies the article's link and pastes it into the detector. Within seconds, the tool identifies misleading statements and provides reliable sources contradicting the claims. Sarah decides not to share the article and politely suggests considering additional sources. The fake news detector helps her make informed decisions and prevents the spread of misinformation.

**Output:**

0.9520258236865539

Text: Celebrities secretly funding underground cult

Prediction: Fake

Text: The Supreme Court on June 19 agreed to urgently hear petitions filed by the West Bengal Government and the State Election Commission against a Calcutta High Court order on June 15 to deploy Central security forces within 48 hours ahead of the panchayat elections in the State.

Prediction: True

**Program Explanation:**

1. Import the required libraries: numpy, pandas, sklearn.naive\_bayes.MultinomialNB, sklearn.feature\_extraction.text.CountVectorizer, sklearn.metrics.accuracy\_score, and sklearn.model\_selection.train\_test\_split.
2. Read the CSV files "True.csv" and "Fake.csv" using pandas' read\_csv() function, and store them in the variables true and fake respectively.
3. Add a "label" column to both the true and fake DataFrames, with a value of 1 for true and 0 for fake.
4. Concatenate the true and fake DataFrames into a single DataFrame called data using pandas' concat() function. Ignore the index to ensure a continuous index for the concatenated DataFrame.
5. Extract the "text" column from the data DataFrame and store it in the variable X.
6. Extract the "label" column from the data DataFrame and store it in the variable y.
7. Split the data into training and testing sets using the train\_test\_split() function. Assign 80% of the data to training and 20% to testing. Set the random\_state parameter to 42 for reproducibility. Store the resulting splits in X\_train, X\_test, y\_train, and y\_test.
8. Initialize a CountVectorizer object called vectorizer.
9. Fit the vectorizer to the training data using the fit\_transform() method on X\_train. This will learn the vocabulary and transform the training text data into a vector representation. Store the transformed training data in X\_train\_vectors.
10. Transform the testing data using the transform() method on X\_test. Store the transformed testing data in X\_test\_vectors.
11. Fit a MultinomialNB classifier to the training data using the fit() method on the classifier object. This will train the classifier on the training data.
12. Use the trained classifier to make predictions on the testing data using the predict() method. Store the predictions in the variable y\_pred.
13. Calculate the accuracy of the classifier by comparing the predicted labels (y\_pred) with the actual labels (y\_test) using the accuracy\_score() function. Store the accuracy score in the variable accuracy.
14. Print the accuracy score.
15. Define a list of new texts that you want to classify.
16. Transform the new texts into vectors using the transform() method on the vectorizer object. Store the transformed vectors in the variable new\_texts\_vectors.
17. Use the trained classifier to make predictions on the new texts using the predict() method. Store the predictions in the variable predictions.
18. Iterate over the new texts and corresponding predictions, and print the text and the corresponding label ("Fake" if label is 0, "True" otherwise).

USE-CASE DIAGRAM:

Scenario: User Interaction

Actors: An actor is an individual, entity, or external structure that plays a role with our

structure in one or more encounters (actors are usually drawn as UML Stick Figures Use

case diagrams).

