APPROACH- FAKE NEWS DETECTION USING PYTHON AND MACHINE LEARNING

TEAM NAME: TRAILBLAZERS

SHRI VARRSHINI M
Computer Science and engineering
Rajalakshmi Institute of Technology
Chennai
shrivarrshini.m.2021.cse@ritchennai.edu.in

SIVARANJANI R
Computer Science and engineering
Rajalakshmi Institute of Technology
Chennai
sivaranjani.r.2021.cse@ritchennai.edu.in

SHANMUGAPRIYA K
Computer Science and engineering
Rajalakshmi Institute of Technology
Chennai
shanmugapriya.k.2021.cse@ritchennai.edu.in

TRAINING MODEL:

The machine learning model used for training the dataset is Support Vector Machine (SVM.) SVM is a powerful supervised machine learning algorithm used for classification and regression tasks. It is particularly effective in solving complex problems with a clear margin of separation between classes. SVM algorithm aims to find an optimal hyperplane in a high-dimensional feature space that maximally separates the classes. The hyperplane is chosen such that the distance between the hyperplane and the nearest data points (support vectors) of each class is maximized.

APPROACH:

1.**IMPORTING THE NECESSARY LIBRARIES:**

Priorly the necessary python libraries such as pandas, seaborn, matplotlib, tqdm, re, nltk, string, os and sklearn have been imported. Further we import word_tokenize function and stopwords module from the nltk package, svm (Support vector machine) module, train_test_split function, accuracy_score, confusion_matrix, TfidfVectorizer from sklearn library.

- The pandas is a popular library for data manipulation and analysis.
- The matplotlib. pyplot module is used for creating various number of plots and charts.
- The seaborn libraries provide a high-level interface for creating attractive and informative statistical graphics.
- The tqdm module provides a progress bar utility for iterating over iterable objects in Python.
- The re module provides functions for working with regular expressions.
- The nltk library stands for Natural Language Toolkit. It is a popular library for natural language processing tasks, such as tokenization, stemming etc.
    - word_tokenize is used to split text into individual words or tokens.
    - from the nltk. corpus package, the stopwords module removes the commonly used words.
- The os module provides a way to interact with the operating system.
- The **sklearn** offers a wide variety of functions and classes that simplify the process of training and evaluating machine learning models.
    - From the sklearn library, the svm module provides support vector machine algorithms for classification and regression tasks.

- From sklearn.model_selection module, the train_test_split function is commonly used to split a dataset into training and testing subsets for machine learning tasks.
- From the sklearn.metrics module, the accuracy_score and confusion_matrix functions are used to evaluate the performance of machine learning models.
- From sklearn.feature_extraction.text module, TfidfVectorizer class is used to convert a collection of raw text documents into a matrix of TF-IDF features.

2.READING THE DATA FILES:

Then we download the required dataset which contains both true and fake news in the excel format. The true and fake news files have been read separately. In order to distinguish true and fake news we are assigning a label value of 1 to each row in the "label" column of the true data Dataframe. This indicates that these rows correspond to true news articles. Then we assign a label value of 0 to each row in the "label" column of the fake data DataFrame. This indicates that these rows correspond to fake news articles.

We concatenate the true data and fake data DataFrames along the rows to create a single DataFrame named data. By combining the two DataFrames into a single DataFrame data, we create a unified dataset that contains both the true and fake news articles which can be useful for further data analysis, preprocessing, or model training.

3.DATA PREPROCESSING:

In the data preprocessing process, we combine the 'title' and 'text' columns into a single 'text' column. Then we remove all the unnecessary columns such as 'title', 'subject', and 'date' which lets us focus only on the combined text.

Then we check the number of missing values (null values) in each column of the DataFrame 'data'. If missing values are present then they can be removed.

We shuffle the rows of the DataFrame 'data' randomly to introduce randomness and avoid any potential bias in the data. By randomizing the order of the rows, it will be useful when splitting the data into training and testing sets or when feeding the data into a machine learning model to prevent any biases introduced by the original order of the data.

4.DATA CLEANING:

We create a function to preprocess the text data. The preprocess function takes a text as input and performs several preprocessing steps on it

- We convert the entire text into lowercase using text.lower() to ensure consistency and to avoid treating words with different cases as distinct.
- We remove all the digits from the text and replace it with an empty string using re.sub() function as the presence of numeric values is not relevant to the analysis and the digits could potentially introduce noise or unwanted patterns.
- Then we remove special characters and punctuation marks from the text and replace it with an empty string using re.sub() function as these symbols are irrelevant to the analysis.
- We remove all the whitespace characters at the beginning and end of the text, effectively trimming the text using the text.strip() function.

- We tokenize the text into individual words using nltk.word_tokenize(text) function. It breaks down the text into meaningful units, allowing further analysis, feature extraction, or model training on the tokenized data.
- We filter out stopwords from the tokenized words focusing on the more meaningful and informative words for analysis and model training.

5.CONVERTING TEXT INTO VECTORS:

The data is split into training and testing sets using train_test_split() function. We assign the test size as 0.2 (i.e) 20% of the data will be used for testing, while 80% will be used for training.

The next step is vectorization which is the process of converting text or other unstructured and data into a numerical feature vectors. The common English stopwords will be ignored during the TF-IDF vectorization process. The maximum document frequency threshold can be set using the max_df parameter. It represents the proportion (0.0 to 1.0) of documents in which a term appears must be considered as a stop word and excluded from the vocabulary. In our case, max_df=0.7 indicates that terms appearing in more than 70% of the documents will be excluded from the vocabulary. fit_transform() on tfidf_vectorizer, it learns the vocabulary from the data and transforms the input text into a TF-IDF matrix representation.

By initializing the TF-IDF vectorizer with stopwords and a maximum document frequency threshold, you can perform TF-IDF vectorization on your text data while considering these parameters to tailor the vectorization process to your needs.

6.MODEL TRAINING:

The model used for training our dataset is Support Vector Machine (SVM) with linear kernel. The linear kernel assumes a linear decision boundary and is commonly used for binary classification tasks. The SVM classifier is trained by svm_classifier.fit() using the TF-IDF vectors from the training data.

The labels for the TF-IDF vectors of the test data are predicted by svm_classifier.predict() using the trained SVM classifier.

After training the SVM classifier, you can use the predicted labels to evaluate the performance of the model, calculate metrics such as accuracy, precision etc.

7. ACCURACY AND CONFUSION MATRIX:

The accuracy of the predicted labels is calculated using accuracy_score() which takes the predicted labels and true labels as parameters. It compares the predicted labels with the true labels and returns the accuracy as a floating point number. The accuracy score represents the proportion of correct predictions. The SVM model we used returns an accuracy of 99.63%

The confusion matrix is generated by confusion_matrix() function which takes true and predicted labels as parameters. The confusion matrix provides insights into the performance of the classification model, showing the counts of true positive, true negative, false positive, and false negative predictions. The confusion matrix produced by the our SVM model is given below

```
[[4685   24]
[   9 4262]]
```

CONTROL FLOW FOR FAKE NEWS DETECTION