# **Fake News Detection Using NLP**

## **Phase – 4**

### In this Phase we did this three steps

# Feature Extraction

# Model Training

# Evaluation

**Feature Extraction:**

It involves tasks such as removing non-alphabetic characters, converting text to lowercase, splitting it into words, removing **stopwords**, and applying **stemming**. These preprocessing steps are essential for preparing text data for classification or other NLP tasks..

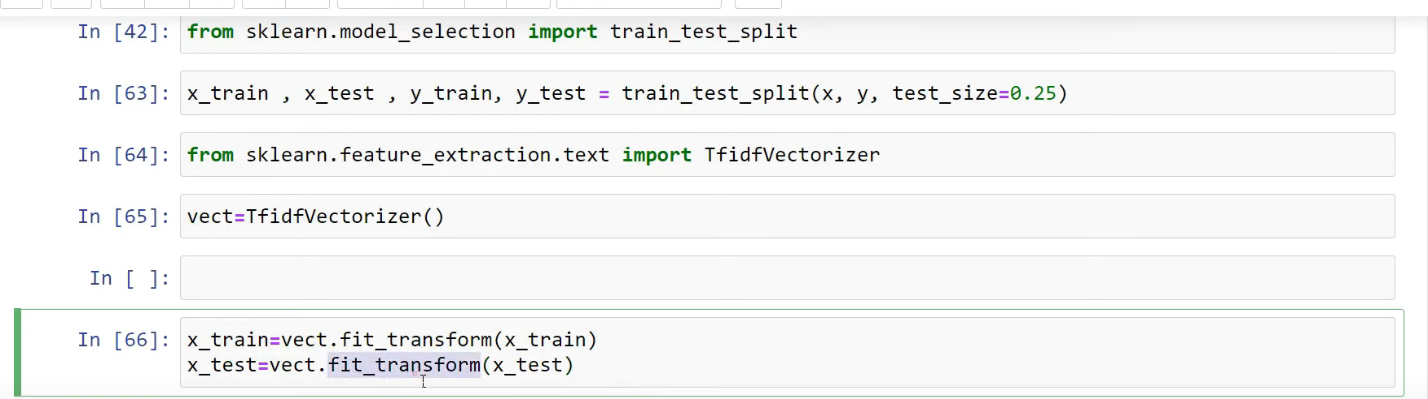




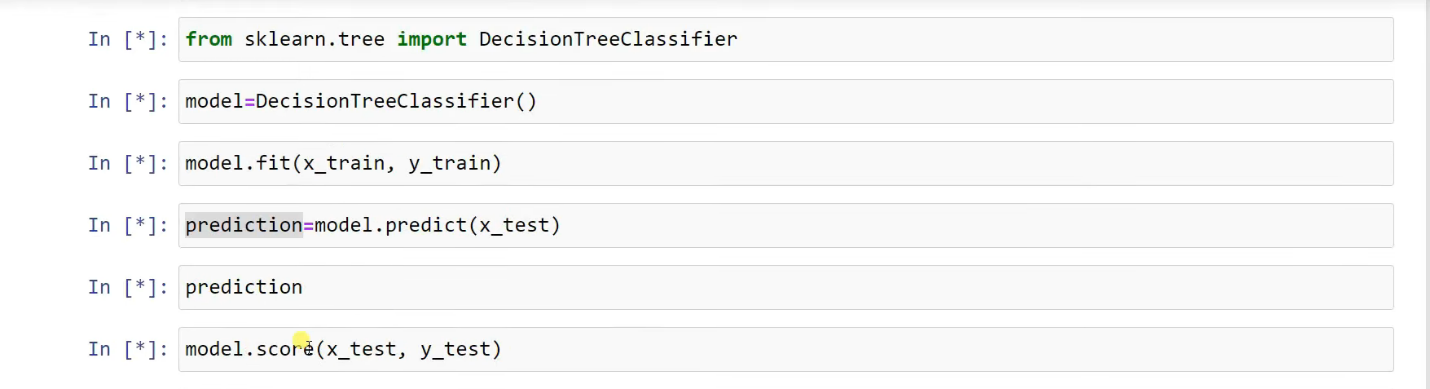
**Model Training:**

Training the Model

1. **train\_test\_split(x, y, test\_size=0.25)**: This function is used to split your dataset x and corresponding labels y into training and testing sets. The test\_size parameter specifies the proportion of the dataset that should be allocated to the testing set (in this case, 25% of the data). It returns x\_train, x\_test, y\_train, and y\_test.
2. **from sklearn.feature\_extraction.text import TfidfVectorizer**: This line imports the **TfidfVectorizer** class from scikit-learn (a popular machine learning library). **TfidfVectorizer** is used for text data preprocessing and feature extraction, particularly for converting text into numerical features using the TF-IDF (Term Frequency-Inverse Document Frequency) method.
3. **vect = TfidfVectorizer():** This line creates an instance of the TfidfVectorizer class. vect will be used to transform your text data into TF-IDF features.
4. **x\_train = vect.fit\_transform(x\_train):** It applies the TF-IDF transformation to the training data x\_train, converting the text data into numerical features. This will replace the original x\_train with the transformed data.
5. **x\_test = vect.fit\_transform(x\_test):** Similarly, it applies the TF-IDF transformation to the testing data x\_test, converting it into numerical features. However, this line should be corrected to use transform instead of fit\_transform. So, it should be x\_test = vect.transform(x\_test) to ensure that the testing data is transformed using the same TF-IDF settings as the training data.



**Evaluation:**



1. **from sklearn.tree import DecisionTreeClassifier**: This line imports the Decision Tree Classifier class from scikit-learn. A decision tree classifier is a type of machine learning model that makes decisions by learning simple decision rules inferred from the training data.
2. **model = DecisionTreeClassifier():** This creates an instance of the Decision Tree Classifier model.
3. **model.fit(x\_train, y\_train):** It trains the Decision Tree Classifier using the training data x\_train and corresponding labels y\_train. The model learns to make predictions based on this training data.
4. **prediction = model.predict(x\_test):** This line makes predictions on the testing data x\_test using the trained decision tree model and stores the predictions in the prediction variable.
5. **model.score(x\_test, y\_test):** This calculates the accuracy of the trained model on the testing data. It compares the model's predictions (based on x\_test) to the actual labels y\_test and returns the accuracy score, which represents the proportion of correctly classified instances in the testing data.

**Team:**

**S.Dhanushiya**

**K.Jelsiya**

**J.Narmatha**

**J.Vijaya**