**Code Explanation**

**Modules Used**

1. NLTK
2. SCIKIT-LEARN
3. TKINTER

Each individual library has its own use, the modules were divided into the team members for their implementation.

**Techniques/ Algorithm used:**

1. **Lemmatization** uses *WordNetLemmatizer* for its implementation.

Lemmatization means converting words into their root form like “*rocks”* is converted to “*rock”*, “*better”* is converted into “*good”,* etc. It uses a dictionary based approach for conversion.

1. **Stemming** is a normalization technique which is much similar to lemmatization except for the fact that it removes all uncommon characters from two words.

E.g. *Entitling*, *Entitled* to *Entitl*

1. **Tokenization** is the method to break paragraphs in list of sentences and then sentences into list of words. This also involves removal of stop words.
2. **Stop Words** are commonly used words which has no meaning in sentence formation. These words normally include articles and helping verbs. E.g. *a, an, the, these, they, his, her, will, was, were, etc.*
3. **TfidfVectorizer** is the main tool which understands which understands and interprets dataset according to the frequency of occurrence of words.

TF-IDF is divided into:

1. *TF:* [Term frequency]
2. *IDF:* [Inverse Document Frequency]

The formula for TF is given by:

*TF(x) = (Frequency of a particular word (x) / Total word count)*

TF decides importance of words in terms of their frequency of words i.e., the more the occurrence of word in document, the more is its weight.

Some words which are not much important may appear more frequently in the dataset.

To overcome the problem IDF is used; as it uses other factors for calculating the importance of words.

The formula is:

*IDF(x) =*

*( log10 Total number of datasets/ Number of datasets with term x in it)*

The overall result is:

*TF – IDF*

Which gives more accurate result.

**Methodology**

1. We have two file “*good\_news.txt” and “bad\_news.txt”.*
2. The sentences from both the files are stemmed, lemmatized and tokenized before analysis.
3. User enters a news headline from the GUI.
4. The news headline is also lemmatized and tokenized.
5. The news headline is compared with tokenized sentences and words of both *good\_news and bad\_news files.*
6. After comparision the probability of similarity is returned.
7. If the probability of similarity of good\_news is more the news is regarded as a Good News else it is regarded as bad\_news.